

ABSTRACT

NI, HAOQI. Online Distributed Control Designs for Cloud-based Wide-Area Power Systems. (Under the direction of Aranya Chakraborty.)

With the increasing complexity in the interconnection of power networks, stability of the system has been studied extensively. One of the biggest challenges for wide-area control of power systems is the need for a highly robust communication system that works in sync with the control functionalities. The majority of the ongoing NASPInet research are devoted to the hardware and architectural planning aspects of wide-area communication, with little attention to understanding how complicated Multiple-Input-Multiple-Output (MIMO) control loops, when implemented on top of the existing communication protocols, may perform under various operating uncertainties. For example, the exponentially increasing number of Phasor Measurement Units (PMUs), producing Terabytes of real-time data that need to be communicated from remote locations to the control centers, may cause a high possibility of network congestion. The network delays in the wide-area communication, however, may cause the closed-loop performance of the grid to degrade. To address the above challenges, in our first study, we propose the adoption of a simple latency control algorithm implemented in a Software-Defined Network (SDN) connecting the VMs in the wide-area cloud by which delays in the control loop can always be optimized in order to retain their stability limit. The algorithm uses a greedy routing path selection based on active latency measurement for data transfer, and can be run periodically to pick the best available routes while the wide-area control loop is running.

Considering perfect delay bounds can almost never be guaranteed no matter how robust the SDN controllers are, in which case the best option might be to drop that link from the control loop while still ensuring that the closed-loop performance remains close

to optimal. In our second study, we present a sparsity-constrained LQR algorithm that starts from an ideal all-to-all connected communication network, and progressively drops links if the traffic in those links are monitored to be high. Each drop is accompanied with a simultaneous optimal tuning of the control gains associated with the existing links using a method called Geromel's algorithm.

However, the more links dropped, the worse the control performance is. In practical situation, the network delay is not constant but time-varying. Therefore, it is obviously not optimal if keep dropping links but never bringing back the dropped links even if the network recovers, which limits the control performance. To address this issue, in our third study, we propose a Kernel Density Estimation (KDE) based model to dynamically detect adverse network delay area by monitoring the round-trip delay of interconnected links. Once a link is detected entering or leaving the delay area, we simultaneously tune the control gains associated with the existing links so that the overall closed-loop performance of the wide-area control system remains approximated optimal. We illustrate the proposed method using the real network data collected from ExoGENI cloud platform and simulations of the IEEE 39-bus power system model. Experimental results show that our framework can effectively guarantee the closed-loop performance for wide-area control in the time-varying network traffics.

In practice, tuning the control gains following delay detection has a very high computing requirement for the control center. If the control center cannot compute the feedback gains within the stability limit time, the system could be unstable as well. Hence, in our fourth study, we present a novel delay aware prediction control framework to address the problem of missing power system state variables due to the existence of communication latency in wide-area measurement control systems. This capability is particularly essential for dynamic power system scenarios where fast remedial control actions are required due to

system events or faults. In this work, a PMU data prediction framework and its practical implementation in power system damping control are proposed to predict future states using the existing PMU data and to complete missing variables with the predicted data. The framework establishes an online prediction scheme, and the proposed implementation adopts recent machine learning advances in data processing. Simulation results indicate that the proposed framework has superior accuracy and is computational efficient, and fulfills the control requirements for power system under delayed network condition.

In our future work, considering the increasing developments of modern power systems, huge amount of data can be obtained from many areas such as SCADA, PUMs. Thus, we propose to further investigate big data and machine learning techniques and tools to help address some critical issues like usability, accuracy and confidence. We will also develop the applications for power system analytics in fault detection and on-line, real-time Dynamic Security Assessment (DSA). In addition, we plan to present the applications for power system operation on outage management and cyber security.

© Copyright 2020 by Haoqi Ni

All Rights Reserved

Online Distributed Control Designs for Cloud-based Wide-Area Power Systems

by
Haoqi Ni

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2020

APPROVED BY:

Dror Baron

Yufeng Xin

Shih-Chun Lin

Aranya Chakraborty
Chair of Advisory Committee

DEDICATION

To my parents and my husband, for their endless support, encouragement and love.

BIOGRAPHY

Haoqi Ni was born in Yangzhou, Jiangsu, China in 1990. She received her bachelor and master degrees in engineering from Shanghai University, China, in 2012 and 2015, respectively. In August 2016, she began her Ph.D. in Electrical & Computer Engineering at North Carolina State University, under the guidance of Dr. Aranya Chakraborty. Her research interests include constrained optimization, software defined network, wide-area control of power system networks, machine learning and applied data science.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support, understanding, encouragement and guidance from many people in my graduate life.

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Aranya Chakraborty, for the continuous support of my Ph.D. study and research, for his wisdom, enthusiasm, and academic guidance during every critical step I took toward becoming an independent researcher. Next, I would also like to thank my committee members Dr. Dror Baron, Dr. Yufeng Xin and Dr. Shih-Chun Lin for their valuable time and comments that improved the presentation and contents of this dissertation.

Additionally, I am thankful to Dr. Yufeng Xin for sharing his wealth of knowledge in networking and machine learning with me. I would also like to thank labmates and friends Jianhua Zhang, Meng Yao, Mang Liao, Sayak Mukherjee, Nandini Negi, Pratishtha Shukla, Hui Yu, Fuhong Xie, Xiaochu Wang and Ming Liang for their helpful guidance, friendship during my graduate life.

Finally, I would especially like to thank my family for their endless love and support over the years. My husband, Ruixin Yang has been extremely supportive of me throughout this entire process and finally help me get to this point. And I would express a deep sense of gratitude to my parents for their continued support and encouragement, without whom I would never have enjoyed so many opportunities.

TABLE OF CONTENTS

List of Tables	vii
List of Figures	viii
Chapter 1 INTRODUCTION	1
1.1 Background	1
1.2 Related works	5
1.2.1 PMU and WAMS	5
1.2.2 Analysis of time delay effect	7
1.2.3 Network communication technology	7
1.2.4 Kernel Density Estimation	8
1.2.5 Long Short Term Memory Time Series Forecasting	9
1.3 Contributions	10
Chapter 2 A Distributed Cloud-based Wide-Area Controller with SDN-Enabled Delay Optimization	11
2.1 Introduction	12
2.2 Wide-Area Control in Presence of Delays	13
2.3 Delay Optimization via SDN Control	17
2.4 ExoGENI-WAMS Cloud Testbed	20
2.5 Wide-area control in ExoGENI with delay optimization	25
2.6 Conclusions	26
Chapter 3 Online Tuning of Cloud-based Wide-Area Controllers with Variations in Network Traffic	27
3.1 Introduction	28
3.2 Control Objective	30
3.3 Structurally Constrained Optimal Control	31
3.3.1 Construction of Feedback Structure	31
3.3.2 Discrete Linear Quadratic Regulator	33
3.3.3 Generalized Riccati Equation Method	34
3.3.4 Algorithm Solution of \mathbb{P}	35
3.4 Implementation on ExoGENI-WAMS Cloud Testbed	36
3.5 Simulation Results	38
3.6 Conclusions	42
Chapter 4 Online Adaptive Wide-area Damping Controller Based on Dynamic Delay Area Detection Using Kernel Density Estimation	44
4.1 Introduction	45

4.2	Online Adverse Delay Area Detection via Machine Learning Method	46
4.2.1	Data Collection	46
4.2.2	Invalid Time Series Forecasting Model for Delay Prediction	47
4.3	Construction of Sparse Control Structure	51
4.4	Simulation Results	54
4.5	Conclusions.	57
Chapter 5 Delay Aware Distributed Data-Driven Control Method for Wide-Area Power Systems		58
5.1	Introduction	59
5.2	Delay Aware Prediction Control Framework	60
5.3	System Implementation	61
5.3.1	Fuzzy Time Series	63
5.3.2	Fuzzy Time Series Forecasting Model	64
5.3.3	Long Short Terms Memory	65
5.3.4	The Workflow of LSTM Predictor	67
5.4	Simulation and Experiments	68
5.4.1	Data Generation	68
5.4.2	Parameter Study of LSTM	69
5.4.3	Prediction accuracy and training time comparison	72
5.4.4	Control performance validation and comparison	76
5.5	Conclusions	84
Chapter 6 Conclusions and Future Work		85
6.1	Conclusions	85
6.2	Future Work	88
BIBLIOGRAPHY		89

LIST OF TABLES

Table 2.1	A summary of experiments to demonstrate an average end-to-end delay analysis over a period of 10 minutes long communication, and time for computing new paths for 300 ms latency injection using <i>nsdelay</i> . In this scenario all nodes communicate data to node <i>N8</i> . . .	20
Table 2.2	Latency between VMs of different racks in USA and Netherlands. . . .	22
Table 2.3	The top three paths were calculated where node <i>N4</i> transmits traffic to <i>N8</i> . <i>L1</i> and <i>L8</i> are stated UNKNOWN as they represent edge links whose latency cannot be determined as the SDN Controller has no communication to end nodes.	24
Table 3.1	Construction of Ω for a 4-VM example	32
Table 3.2	Network status and Value of <i>J</i>	42
Table 4.1	The comparison of RMSE for different ARIMA forecasting steps	49
Table 4.2	3 Cases for validating the control performance	55
Table 5.1	Two-sample T-Test of Validation Loos for Gen 1 based on Different Hidden Nodes	70
Table 5.2	Two-sample T-Test of Validation Loos for Gen 2 based on Different Hidden Nodes	71
Table 5.3	Two-sample T-Test of Validation Loos for Gen 5 based on Different Hidden Nodes	72
Table 5.4	Two-sample T-Test of Validation Loos for Gen 10 based on Different Hidden Nodes	72
Table 5.5	The comparison of RMSE for different models against actual values .	74
Table 5.6	Average Time of Training and Prediction for Different LSTM Hidden Nodes and Fuzzy Model	76
Table 5.7	6 Cases for validating the control performance	77

LIST OF FIGURES

Figure 1.1	PMU deployment as of March 2012 [38]	6
Figure 1.2	Structure of WAMS	6
Figure 2.1	Flowchart of wide-area delay-aware controller	15
Figure 2.2	System instability when the delay is greater than threshold	16
Figure 2.3	Performance evaluation of the optimized and non-optimized SDN controller under injection of 300 ms latency.	19
Figure 2.4	Architecture of the ExoGENI-WAMS Testbed	21
Figure 2.5	Topological representation of the WAMS delay-aware control with latency optimization constructed on ExoGENI testbed.	23
Figure 2.6	IEEE 39-bus New England power system model	23
Figure 2.7	Generator state responses under optimized delay control	25
Figure 3.1	Architecture of the ExoGENI-WAMS Testbed	37
Figure 3.2	Network topology of different situations	40
Figure 3.3	Performance of different controllers in damping oscillations	41
Figure 3.4	Performance of sparse control algorithm	43
Figure 4.1	Example of RTT variations between RENC I (Chapel Hill, NC USA) and OSF (Oak-land, CA USA) on ExoGENI cloud platform	47
Figure 4.2	Time series of the RTT observations and the forecasts provided by ARIMA model for different steps ahead	48
Figure 4.3	Flowchart for Delay Area Detection algorithm by KDE-based Model	51
Figure 4.4	Adverse Delay Area Detected by the Proposed KDE-based Model . . .	52
Figure 4.5	Example of network conditions for 3 bi-directional links	53
Figure 4.6	Performance of different cases in damping oscillations	56
Figure 4.7	Output response for all generators	56
Figure 5.1	The typology of the DAPC	61
Figure 5.2	The example of DAPC states completion	62
Figure 5.3	Structure of LSTM: A cell of LSTM.	66
Figure 5.4	Structure of LSTM: Unrolled form of LSTM.	66
Figure 5.5	LSTM Loss of Gen 1: (a) Training loss. (b) Validation loss.	70
Figure 5.6	LSTM Loss of Gen 2: (a) Training loss. (b) Validation loss.	70
Figure 5.7	LSTM Loss of Gen 5: (a) Training loss. (b) Validation loss.	71
Figure 5.8	LSTM Loss of Gen 10: (a) Training loss. (b) Validation loss.	71
Figure 5.9	Prediction Steps Clarification.	73

Figure 5.10 Prediction accuracy comparison 75
Figure 5.11 Performance of Case 1 in damping oscillations 78
Figure 5.12 Performance of Case 2 in damping oscillations 79
Figure 5.13 Performance of Case 3 in damping oscillations 80
Figure 5.14 Performance of Case 4 in damping oscillations 81
Figure 5.15 Performance of Case 5 in damping oscillations 82
Figure 5.16 Performance of Case 6 in damping oscillations 83

CHAPTER

1

INTRODUCTION

1.1 Background

In recent years, wide-area damping control (WADC) has been extensively studied to suppress inter-area oscillations in power systems [65]. Since the US Northeast blackout of 2003, the fast development of wide-area measurement system (WAMS) brings new chances to the damping control of inter-area power systems. WAMS leads utility owners to understand how the interconnected nature of the grid topology essentially couples their controller performance with that of others, and thereby forced them to look beyond this myopic approach

of local feedback and instead use wide-area measurement feedback. A survey presented in [38] shows that the deployment of wide-area monitoring and control (WAMC) system is of the highest priority among the Nordic transmission grid operators (TSOs). Meanwhile, North America Synchronphasor Initiative (NAPSI) has also identified WAMC as an enabler for the smart transmission grid being deployed in North America [41]. During the past decades, several efforts have been directed towards WAMC based on robust and optimal control methods. Liu et al. [33] applied the linear parameter varying controller design technique in the design of a supplementary damping controller for a thyristor controlled series capacitor. Tomsovic et al. [56] provided several example novel control and communication regimes for such as a complete redesign of the control, communication and computation infrastructure. Chakraborty [7] presented a flexible AC transmission systems-based control design for electromechanical oscillation damping in large power systems. Zima et al. [70] discussed the basic design and special applications of wide-area monitoring and control systems, which complement classical protection systems and supervisory control and data acquisition/energy management system applications. Boukarim et al. [4] used several control design techniques to coordinate two power system stabilizers to stabilize a 5-machine equivalent of the South/Southeast Brazilian system. Zhang et al. [69] developed a systematic procedure of designing a centralized damping control system for power grid interarea oscillations putting emphasis on the signal selection and control system structure assignment. Chaudhuri et al. [9] designed a WAMS based damping controller for a prototype power system using a static var compensator. A tutorial on the ongoing practices for wide-area control has recently been presented in [6], while cyber-physical implementation architectures for realizing these controls have been proposed in [1], [54].

One of the biggest challenges for a wide-area control infrastructure is the need for a highly robust communication system that works in sync with the control functional-

ities. The majority of the ongoing NASPInet research are devoted to the hardware and architectural planning aspects of wide-area communication [38], with little attention to understanding how complicated multiple-input-multiple-output (MIMO) control loops, when implemented on top of this communication, may perform under various operating uncertainties. The most critical uncertainty is delay arising from packet communication (queuing and transport) latency in the network and from computation overheads in the application servers. With the exponentially increasing number of Phasor Measurement Units (PMUs), producing Terabytes of real-time data that need to be communicated from remote locations to the control centers, the problem of such collective delays is becoming significant. The primary reason why delay is anticipated to become an unavoidable challenge for wide-area control is because utilities are unlikely to establish dedicated communication links for these types of controls, which means that the communication infrastructure must be implemented on top of their existing networks. As a result, PMU data used for control will have to be transported over a shared network, sharing bandwidth with other ongoing applications, giving rise to significant delays due to queuing and routing in addition to transport. As analyzed in [3], [39], [59], the current communication infrastructure of wide-area measuring system (WAMS) cannot guarantee a satisfactory QoS. The resulting data quality issue like data accuracy, data loss and data latency can significantly impact the system control performance. This problem is widely recognized in the previous literature, see [20], [31], [39], [40], [45], [64] for examples.

In recent literature, several researchers have looked into delay mitigation in wide-area control loops [10], [68]. All of these designs are, however, based on fixed and worst-case delays, which makes the controller unnecessarily restrictive, and may degrade closed-loop performance. In the recent paper [53], this problem was addressed by proposing a delay-aware wide-area controller, where the feedback gain matrix was made an explicit function of

delays using ideas from arbitrated network control theory. The controller was implemented using a cloud computing environment with virtual computers located inside local clouds of each utility company. The limitation of that design, however, was that the delays were left unmonitored and uncontrolled, and hence for real-time implementation there was no guarantee that the delays may not violate their assumed upper bounds.

Therefore, in this work, we focus on developing wide-area communication and control from a Cyber-Physical System (CPS) viewpoint. We study the ways in which these practices pose limitations against optimal grid performance, how modern communication technologies such as Software-Defined Networking (SDN) can be used to overcome these limitations, and how sparse control algorithm can be integrated with the network delay and uncertainties of these wide-area networks to ensure that the closed-loop grid operates in a stable, reliable, robust, and efficient way. We also discuss the use of machine learning method such as Kernel Density Estimation (KDE) to detect the adverse delay area and Long Short Term Memory (LSTM) to predict the power system states unreceived because of network latency or packet loss, thus ensure the control performance of wide-area controllers.

In our future work, considering the increasing developments of modern power systems, huge amount of data can be obtained from many areas such as SCADA, PUMs. Thus, we propose to further investigate big data and machine learning techniques and tools to help address some critical issues like usability, accuracy and confidence. We will also develop the applications for power system analytics in fault detection and on-line, real-time dynamic security assessment (DSA). In addition, we will present the applications for power system operation on outage management and cyber security.

1.2 Related works

1.2.1 PMU and WAMS

A phasor measurement unit (PMU) is a device used to estimate the magnitude and phase angle of time-varying electrical phasor quantities (such as voltage or current) in the electricity grid using a common time source provided by a global positioning system (GPS) radio clock. It allows synchronized real-time measurements of multiple remote points on the grid. PMUs are capable of capturing samples from a waveform in quick succession (6-60 samples/sec) and reconstructing the phasor quantity, made up of an angle measurement and a magnitude measurement. The resulting measurement is known as a synchrophasor. This map in Fig. 1.1 shows a snapshot of PMU deployment as of March 2012.

Based on the synchronous phasor measurement and the modern communication technology, WAMS is a collective technology to monitor power system dynamics in real time, identifies system stability related weakness and helps to design and implement counter measures. The main components of WAMS are PMUs, PDC (Phasor Data Concentrator), GPS, communication channel that connecting them, with a structure shown in Fig. 1.2. There search of WAMS started from 1995 by the U.S. Department of Energy to develop advanced tools for wide-area measurement, control, and operation in the Western North American power system (WECC). Since WAMS is able to accurately provide both individual and sequential voltage and current phasors, it has been extensively used in power system applications, such as power system monitoring [5], power system state estimation [58], power system protection [47], and power system control [66].

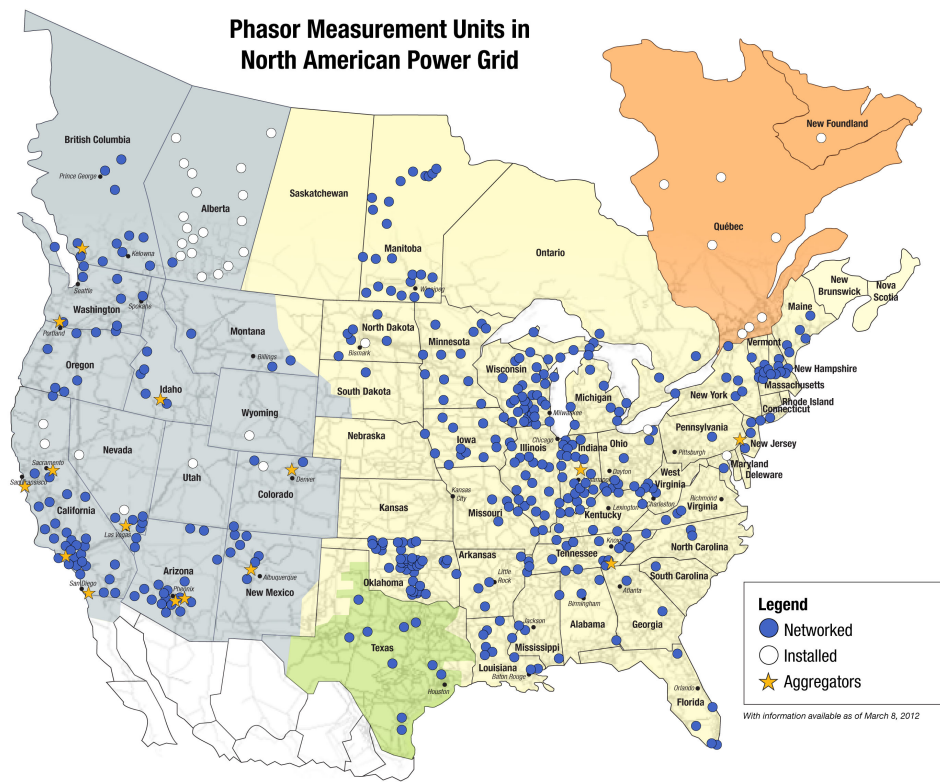


Figure 1.1 PMU deployment as of March 2012 [38]

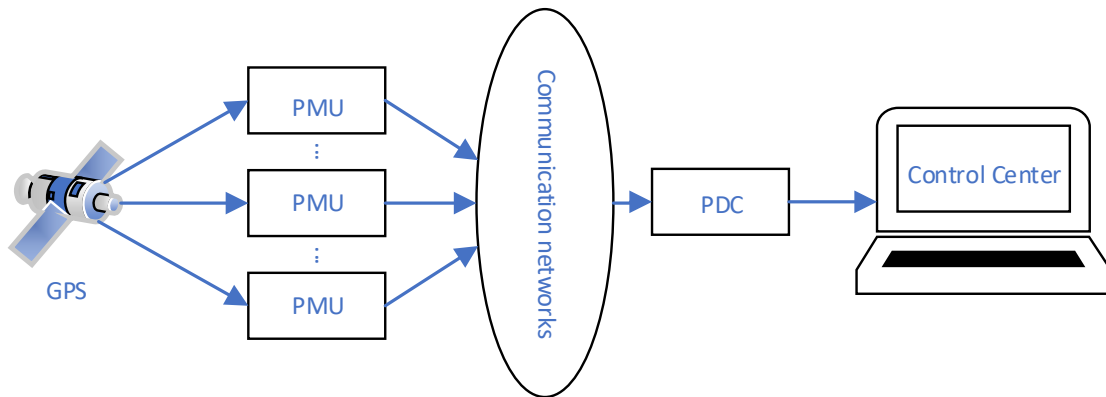


Figure 1.2 Structure of WAMS

1.2.2 Analysis of time delay effect

The emergence of WADC has brought out many communication problems as wide area feedback signals are transferred through communication network, such as time delay, data loss, disordering and communication failure, which resulting in deterioration of control performance and system stability. Such challenge is evidenced by a recent report from NASPI stating that 71% of the networked PMUs in the US power grid on the east coast failed to deliver valid data because of issues associated with communication networks [35]. In [55], time delays can ranging from 0.02 s to 0.5 s that are affected by transmission channels were reported.

A time varying phase lag module introduced by time delay in control system may lead the WADC system to be unstable. Thus, delay must be considered in the control design stage to improve the controller performance. Delay compensation is widely used in WADC, such as Padé approximations [29], Smith predictor [62] and phase compensation with lead-lag filters [67]. These methods only consider the time delay to be a constant value. However, the time delay is stochastic in practice. Therefore, it is necessary to consider the stochastic time delay in the design. One approach is robust control, in which LMI method is often used to calculate controller parameter [61]. Another approach is to measure time delay from timestamp of each data packet and to compensate stochastic time delay continuously [11]. However, detailed stochastic distribution of time delay has not been considered in any past methods, which should be considered in future research.

1.2.3 Network communication technology

As mentioned earlier, typically the Internet cannot provide the required latency and packet loss performance for grid operation under high data rates. And uncontrolled delays in

a network can easily destabilize distributed control algorithms for wide-area oscillation monitoring using PMU data from geographically dispersed locations. Moreover, the network performance is highly random, and therefore, difficult to model accurately. That is why the cloud-based WAMS architecture proposed in Fig. 1.2 is currently garnering a lot of attention from power system engineers. However, limited studies have been conducted so far to leverage all possible benefits of cloud computing, Software-Defined Networking (SDN), and Network Function Virtualization (NFV), to accelerate this development [25]. With the recent revolution in networking technology, these new communication mechanisms can open up several degrees of freedom in programmability and virtualization for computation and communication platforms. According to latest IT industry analysis, the global market for SDN has recorded a remarkable annual growth rate of 88.1% in five years period only. The market has reached 2.4\$ billion in 2015 and is expected to reach 56.1\$ billion in 2020 [21]. SDN has been adopted by dominant internet companies like Google and Facebook for their Data Center Networks structure and their connections in Wide Area Networks [16]. However, customized SDN control and protocols, and sufficient experimental validation using realistic testbeds are still missing in almost all wide-area control applications. In recent years, simulation platforms such as ExoGENI-WAMS [8] have been developed to emulate such communication platforms. The computation and communication planes are entirely shifted away from the physical infrastructure, similar to the architecture proposed in Fig. 1.2.

1.2.4 Kernel Density Estimation

As mentioned in the above, time delay in control system may lead the WADC system to be unstable. It is of great value to estimate probability distributions of network delay to avoid

the influence of such delay. Nonparametric density estimators are particularly useful for their broad applicability in estimating unknown distributions, as they can account for irregular structures without requiring an understanding of the underlying network mechanisms. Kernel density estimation (KDE) is the most statistically efficient nonparametric method for probability density estimation known and is supported by a rich statistical literature that includes many extensions and refinements [23], [48], [57].

1.2.5 Long Short Term Memory Time Series Forecasting

Not only network delay but also packet drop is a problem that cannot be ignored during the transmission. So it's important to predict and recover the missing states for the wide-area damping control system. Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture [22] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as handwriting, speech or video)[19], [30], [44].

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

1.3 Contributions

The main contributions of this thesis are stated as follows:

1. We propose an optimized SDN controller to select a shortest path in transmitting the data and validate the experiments on Exo-GENI testbed.
2. Development of sparse LQR design with structural constraints on the feedback matrix to sparsify the congested communication links.
3. KDE based model to detect abnormal delay which may potentially cause the system unstable.
4. Design of data-driven delay aware prediction control framework to predict and recover the missing states under dynamic networks.

CHAPTER

2

A DISTRIBUTED CLOUD-BASED
WIDE-AREA CONTROLLER WITH
SDN-ENABLED DELAY
OPTIMIZATION

2.1 Introduction

The use of cloud computing for wide-area control of power systems using distributed computational resources has recently garnered significant attention. The upshot in such implementation, however, is that Synchrophasor data from one computational agent, commonly referred to as a virtual machine (VM), located in a certain location in the cloud need to be transmitted to VMs located in other locations through a shared communication network. Due to fluctuating bandwidth of these shared links, the wide-area control action can suffer from unacceptable delays, which may not only degrade closed-loop performance but can also threaten grid stability if they exceed a certain limit. This chapter proposes the adoption of a simple latency control algorithm implemented in a software-defined network (SDN) connecting the VMs in the wide-area cloud by which delays in the control loop can always be optimized in order to retain their stability limit. The algorithm uses a greedy routing path selection based on active latency measurement for data transfer, and can be run periodically to pick the best available routes while the wide-area control loop is running.

This chapter resolves this problem by adopting a latency control algorithm by which delays in the control loop can be maintained within their stability limit. The algorithm uses a greedy routing path selection for data transfer, and can be run periodically to pick the best available link routes while the wide-area control loop is running. Results are validated using the IEEE 39-bus power system model. A cloud computing testbed called ExoGENI is used for implementing the delay control algorithm that controls a network of virtual SDN switches interconnecting the VMs. Experimental results show that the proposed framework can effectively improve the closed-loop performance for wide-area control against the adverse effects of the delays.

2.2 Wide-Area Control in Presence of Delays

Consider a power system network with n synchronous generators. Each generator is modeled by a flux-decay model assuming that the time constants of the d - and q -axis flux are fast enough to neglect their dynamics, that the rotor frequency is around the normalized constant synchronous speed, and that the amortisseur effects are negligible. The model of the i^{th} generator is then written as [28]:

$$\dot{\delta}_i = \omega_i - \omega_s \quad (2.1)$$

$$M_i \dot{\omega}_i = P_{mi} - (V_i I_{qi} \cos(\theta_i - \delta_i) + V_i I_{di} \sin(\delta_i - \theta_i)) - d_i(\omega_i - \omega_s) \quad (2.2)$$

$$T_{qi} \dot{E}'_{qi} = -E'_{qi} + (x_{di} - x'_{di})I_{di} + E_{fdi} \quad (2.3)$$

$$T_{di} \dot{E}'_{di} = -E'_{di} + (x_{qi} - x'_{qi})I_{qi} \quad (2.4)$$

$$T_{Ai} \dot{E}_{fdi} = -E_{fdi} + K_{Ai}(V_{ref,i} - V_i) + u_i(t). \quad (2.5)$$

for $i = 1, \dots, n$. Equations (2.1)-(2.2) are referred to as the swing equations while (2.3)-(2.5) as the excitation equations. The states δ_i , ω_i , E'_{qi} , E'_{di} , and E_{fdi} respectively denote the generator phase angle (radians), rotor velocity, the quadrature-axis internal emf, the direct-axis internal emf, and the field excitation voltage. The voltage at the generator terminal bus is denoted in the polar representation as $\tilde{V}_i(t) = V_i(t) \angle \theta_i(t)$. $V_{ref,i}$ is the constant setpoint for V_i . The generator current in complex phasor form is written as $I_{di} + jI_{qi} = I_i \angle \phi_i$. ω_s is the synchronous frequency, which is equal to 120π rad/sec for a 60-Hz power system. M_i is the generator inertia, d_i is the generator damping, and P_{mi} is the mechanical power input from the i^{th} turbine, all of which are considered to be constant. T_{di} , T_{qi} , and T_{Ai} are

the excitation time constants; K_{Ai} is the constant voltage regulator gain; x_{di} , x'_{di} , x_{qi} and x'_{qi} are the direct-axis and quadrature-axis salient reactances and transient reactances, respectively. All variables, except for the phase angles (radians), are expressed in per unit. The model is linearized across a load flow solution, and the small-signal model is written in a compact form as

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0, \quad x \in \mathbb{R}^{5n}, \quad u \in \mathbb{R}^n \quad (2.6)$$

where the definition of the state vector and excitation input for each generator follow from (2.1)-(2.5), A is the state matrix, B is the control input matrix, and x_0 is the initial state of the system. Assuming a decentralized Kalman-filter based state estimator located at every generator bus as in [49], availability of full state feedback is assumed. A wide-area LQR controller is next designed as:

$$\begin{aligned} \min_K J &= \frac{1}{2} \int_0^\infty (x^T(t)Qx(t) + u^T(t)Ru(t))dt \\ \text{s.t., } \dot{x}(t) &= Ax(t) + Bu(t) \\ u(t) &= K\hat{x}(t), \quad K \in \mathcal{S}, \quad Q > 0, \quad R \geq 0, \end{aligned} \quad (2.7)$$

where the choice of Q and R follows from damping requirement. The control gain matrix K in general can be designed as a structured matrix, but for this chapter no structural constraints are posed for simplicity, i.e., it is assumed that every generator state vector is communicated to every generator control input.

Once designed, the controller (7) is next implemented in a cloud computing environment using spatially distributed virtual computers or virtual machines (VMs), as explained in [53]. The idea for the implementation is as follows. Each generator in the physical network

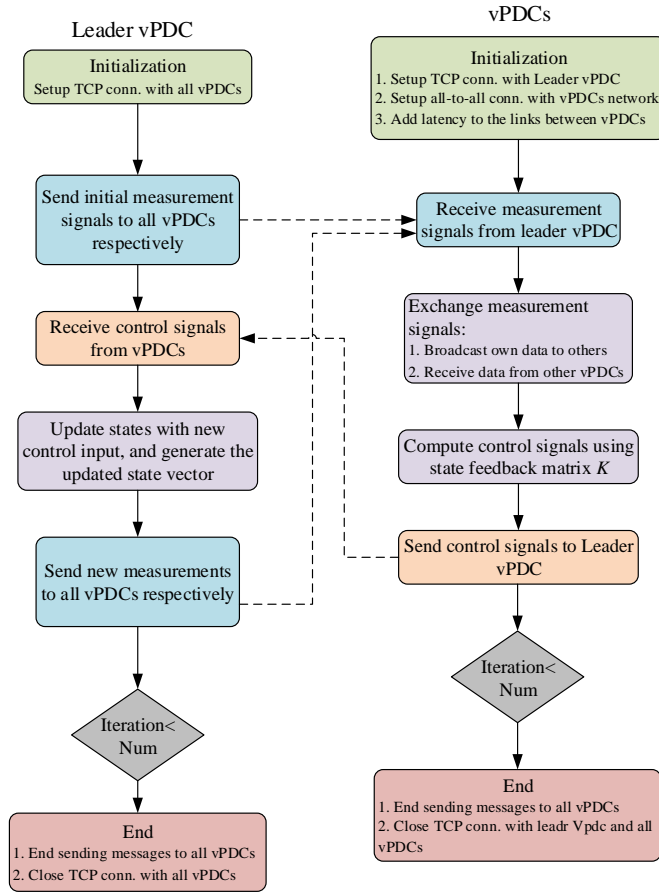


Figure 2.1 Flowchart of wide-area delay-aware controller

is assigned a unique VM in the cloud. The actual geographical location of this VM may be in close vicinity to the location of the generator. State estimate x_i of the i^{th} generator is communicated to its respective VM through a local-area network. The VM for the i^{th} generator contains the corresponding block-row K_i of the control gain matrix K pre-embedded in it. This VM then exchanges $x_i(t)$ with every other VM, so that every VM has access to the entire state vector $x(t)$. Each VM i then computes its respective control signal $u_i = K_i x(t)$, and transmits it back to the physical grid for actuation. However, in practical implementation, the VM-to-VM communication through the wide-area network will always incur delays as

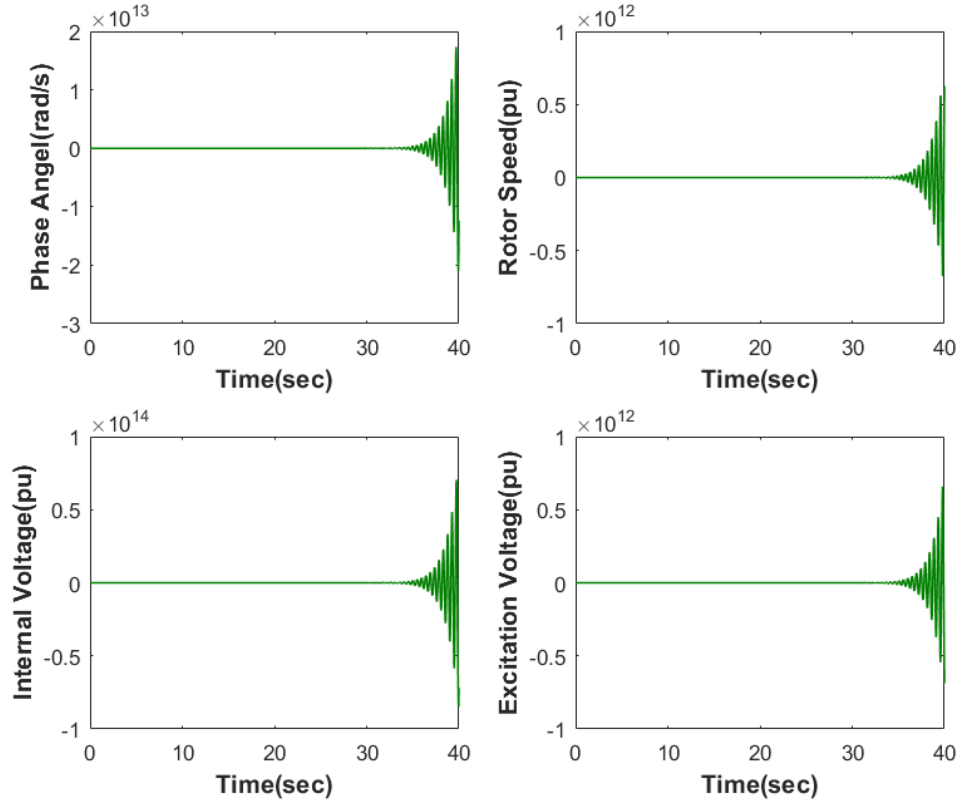


Figure 2.2 System instability when the delay is greater than threshold

a result of which the control input will be of the form

$$u(t) = Kx(t - \tau) \quad (2.8)$$

where $\tau = \max_{ij} \tau_{ij}$ between any two VMs i and j . The delay in every channel is assumed to be fixed at τ in order to maintain synchronous communication. The discrete-time small-signal model (A_d, B_d) for the IEEE 39-bus New England system derived from the discretization of (2.6) using sampling time $T = 33$ ms (or 30 samples per second frequency) is considered for simulations. Depending on A_d and B_d , τ will have different upper bounds for closed-loop stability. The control loop is implemented using 11 VMs in the ExoGENI

network. One of these VMs is assigned to be a *leader* VM, while the remaining ten are followers. The implementation is done in the following way. The leader here emulates the power system model, and generates $x_i(t)$ by running the discrete-time model for (2.6). The followers keep receiving their respective $x_i(t)$ from the leader continuously, and thereafter exchange them between each other. The flow-chart for implementation is shown in Fig. 2.1.

The *nsdelay* module in ExoGENI is used to inject artificial delays to test the bounds for τ in the links between VM1 through VM10. From offline simulations it was observed that if τ exceeds 66 ms then the closed-loop is unstable. Fig. 2.2 shows this instability. To maintain τ within this hard limit, a path optimization method in SDN is implemented by which the routes containing the least congested links in the cloud network can be selected periodically in real-time as the control loop is in operation, thereby attempting to retain the end-to-end delay to be less than 66 ms for all time $t > 0$.

2.3 Delay Optimization via SDN Control

SDN is an emerging network technology that has gained wide deployment. One main advantage of SDN is the centralized controller that allows more efficient application specific network programming and performance optimization. However the existing SDN controllers, for example, the popular Floodlight controller, do not take link delay and available bandwidth conditions into consideration in deciding the end-to-end paths. For this study, a SDN controller that was recently developed in [15] using multiple end-to-end delay calculation techniques (i.e. LLDP, BDDP, statistical methods, etc.) is used to compute and select the optimal paths dynamically. To demonstrate the effectiveness and efficiency of this SDN-based framework, in this section, a comparison with a non-optimized Floodlight SDN controller is also presented. Floodlight in its default configuration allows packet probing.

The use of LLDP and BDDP facilitates the discovery and latency measurement of paths and links in a network. The non-optimized controller removes the procedure to evaluate link latency and treats all links equally while utilizing the discovery processes to determine the network topology. In contrast, the SDN controller used here considers both Timestamp Recording (TR) and shortest path selection method to compare the arrival and departure time of flows and packets over a period of time and to compute the optimal path for selected traffic flows. The calculation of TR is simply given as

$$\text{Delay}(S, R) = t_R - t_S \quad (2.9)$$

$$\text{where } t_R = \frac{1}{n} \sum_{k=1}^n t_R^k, \quad t_S = \frac{1}{n} \sum_{k=1}^n t_S^k. \quad (2.10)$$

Here, t_R and t_S represent the receiving and sending time, respectively.

The effectiveness of the controller is evaluated using *nsdelay* available in ExoGENI. The discussion here relates to a general set-up of eleven SDN nodes in ExoGENI. The actual set-up used for controlling the 39-bus power system will be discussed in the following sections. The purpose here is to force both SDN controllers to recompute the shortest path while observing the network behavior between the eleven nodes. A SDN topology is constructed and implemented in ExoGENI by considering an environment with SDN switching and routing devices, a software module to emulate network traffic and end-hosts. The architectural design of this topology makes use of Open VSwitch (OVS) that supports OpenFlow 1.3. We further select Floodlight 1.2 as our SDN controller since it supports both physical and virtual OpenFlow switches, has the capabilities to handle both OpenFlow and non-OpenFlow networks, and provides QoS features. Figure 2.3 demonstrates a comparison of the non-optimized network behavior versus the optimized version of the Floodlight SDN controller, where Node $N5$ communicates with Node $N9$. The sharp increase in the

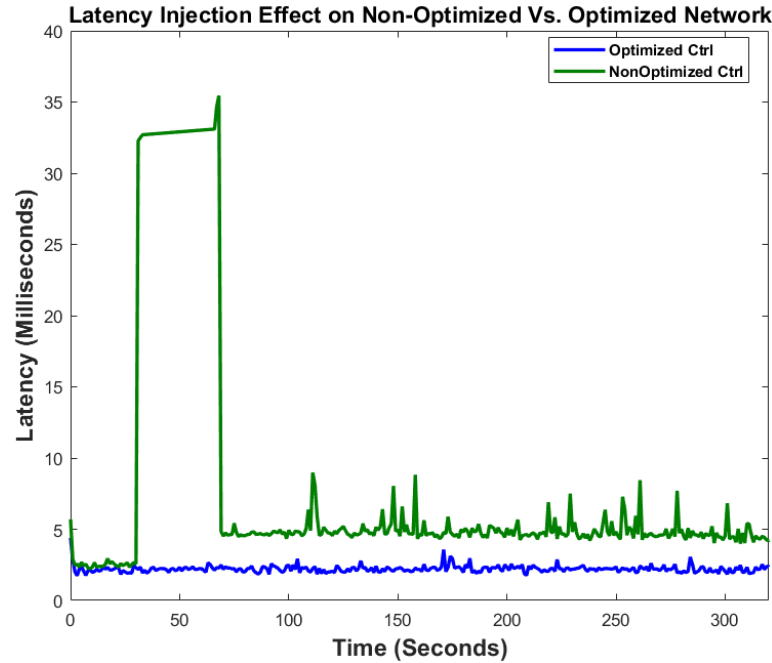


Figure 2.3 Performance evaluation of the optimized and non-optimized SDN controller under injection of 300 ms latency.

green plot represents a significance increase of end-to-end delay in the non-optimized network once 300 ms latency is injected into the communication path between $N5$ and $N9$. After identifying the first optimal path selected by both SDN controllers, latency is injected into the most critical inter-link between $N5$ and $N9$ where both SDN controllers aim to recompute and select an alternative optimal path. However, the operation of reselecting a new path does not impact the end-to-end delay delivered by the optimized SDN. In the non-optimized SDN network, however, the end-to-end delay significantly increases during the path re-selection operation as depicted by Table 2.1.

Once an excessively large latency of 300 ms is injected, the non-optimized SDN controller requires a notable time interval to reselect a new path and redirect the ongoing traffic to the final destination. This leads to a significant end-to-end delay during the time it

takes to reselect an alternative path to redirect the continuous network traffic. This process, however, does not intend to impact the end-to-end delay and data delivery between $N5$ and $N9$.

Table 2.1 A summary of experiments to demonstrate an average end-to-end delay analysis over a period of 10 minutes long communication, and time for computing new paths for 300 ms latency injection using *nsdelay*. In this scenario all nodes communicate data to node $N8$.

Client	Non-Optimized		Optimized	
	Avg Delay	Path Selection	Avg Delay	Path Selection
N1	6.41 ms	4.2 ms	4.05 ms	1.01 ms
N2	5.18 ms	4.19 ms	3.39 ms	0.76 ms
N9	4.23 ms	5.5 ms	3.62 ms	1.13 ms
N11	6.97 ms	5.3 ms	3.17 ms	1.02 ms

Table 2.1 shows the average time taken by both SDN networks to reselect a new optimal path as well as the end-to-end delay results of the non-optimized vs. the optimized network under the impact of *nsdelay* latency injection. In this experimental scenario, Nodes $N1$, $N3$, $N9$ and, $N11$ here all simultaneously communicate with node $N8$. The numbers in this Table 2.1 show that the optimized SDN outperforms the non-optimized network in terms of providing better quality of service, while maintaining the stability limit for the wide-area controller. Additionally, the optimized SDN framework takes negligible amount of time to recompute an alternative optimal path, while the non-optimized controller takes as long as 4 ms to do so.

2.4 ExoGENI-WAMS Cloud Testbed

The implementation of the delay optimizer for the wide-area controller (8) is next demonstrated using the ExoGENI-WAMS testbed [18]. This testbed consists of two parts: (1) The physical layer consists of the power system model implemented in real-time digital sim-

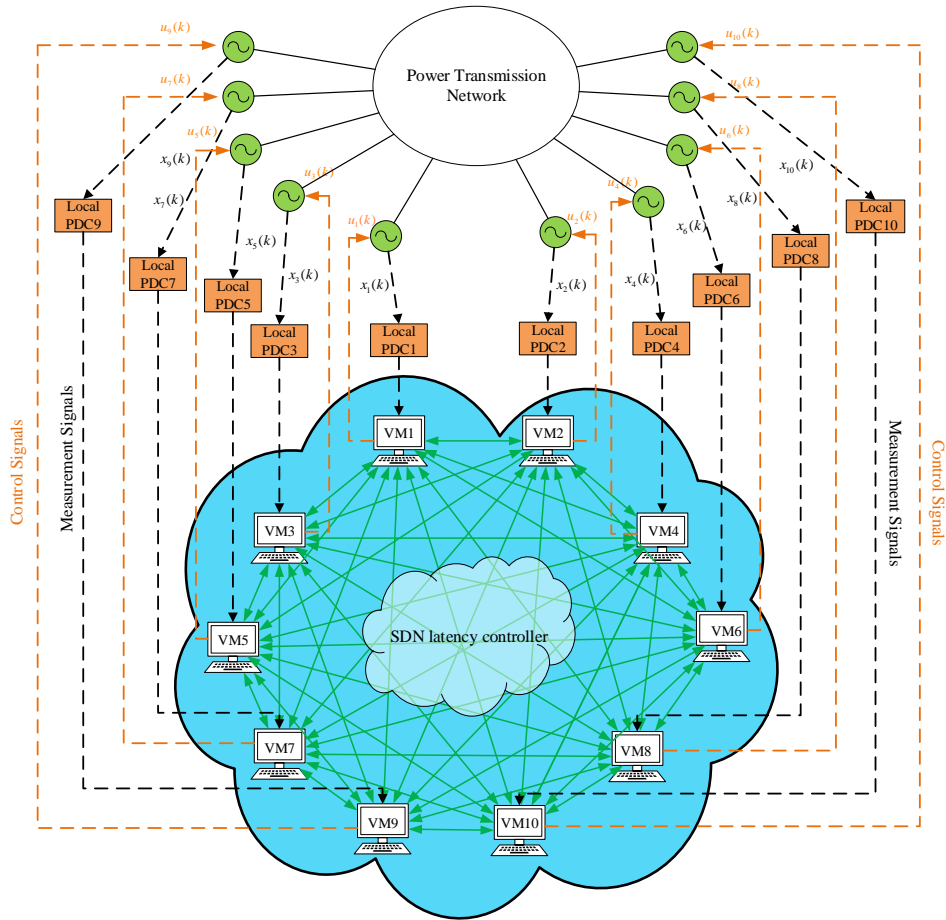


Figure 2.4 Architecture of the ExoGENI-WAMS Testbed

ulators (RTDS) and hardware PMUs generating measured signals from RTDS. Both are equipped with network cards to communicate with the cloud. (2) The cyber layer is represented by the ExoGENI distributed cloud environment which connects to the physical system via high bandwidth virtual network links. It has several dozens of Cloud sites geographically distributed over the world that can provide on-demand virtual topologies of different scales and capabilities. The virtual topology is created with customized virtual machines (VMs) interconnected by virtual network links. ExoGENI provides both graphical interface (ORCA-Flukes) and programmable APIs for creating and managing virtual

Table 2.2 Latency between VMs of different racks in USA and Netherlands.

From	To	Avg Latency
RENCI (Chapel Hill)	RENCI (Chapel Hill)	0.643 ms
RENCI (Chapel Hill)	BBN/GPO (Boston, MA)	16.998 ms
CIENA (Ottawa, CA)	RENCI (Chapel Hill)	38.897 ms
CIENA (Ottawa, CA)	UvA (Amsterdam)	88.068 ms
UAF (Fairbanks, AK)	UvA (Amsterdam)	210.774 ms

topology.

Fig. 2.4 illustrates this cyber-physical architecture that implements the proposed cloud-in-the-loop wide-area controller (2.6)-(2.8). In this figure, PMUs are installed at every generator bus. The PMU data from generators in a specific area in the grid are aggregated at a local control center, denoted as a local PDC. The data are then communicated to the control application hosted in the Cloud. Each generator is only allowed to communicate with its own local control server implemented in a VM in its designated Cloud site. For the sake of the experiments here, a leader VM is used to generate the PMU data and receive control signals, thereby emulating the functions of generators and local PDCs shown in Fig. 2.4. The leader VM generates the measurement signals at every sample time and broadcasts them to all the local VMs. Upon receiving the data, the local VMs exchange the information they received, compute the control signals and send back to leader VM progressing the generator models, and getting measurements for the next sampling instant. The communication between leader VM and each local control VM in practice will occur over a private network, for which the delay is negligible. However, the communications between the local VMs will incur large random delays as they are distributed over a wide-area network, decided collectively by the VM topology, physical locations of VMs, and the background network traffic. Table 2.2 shows the average network latency between different ExoGENI sites measured in the experiment. From the table, it can be seen that the latency

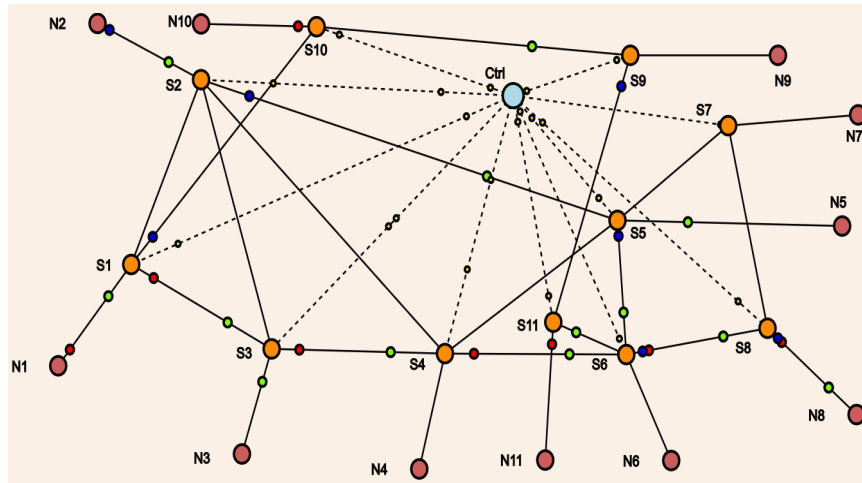


Figure 2.5 Topological representation of the WAMS delay-aware control with latency optimization constructed on ExoGENI testbed.

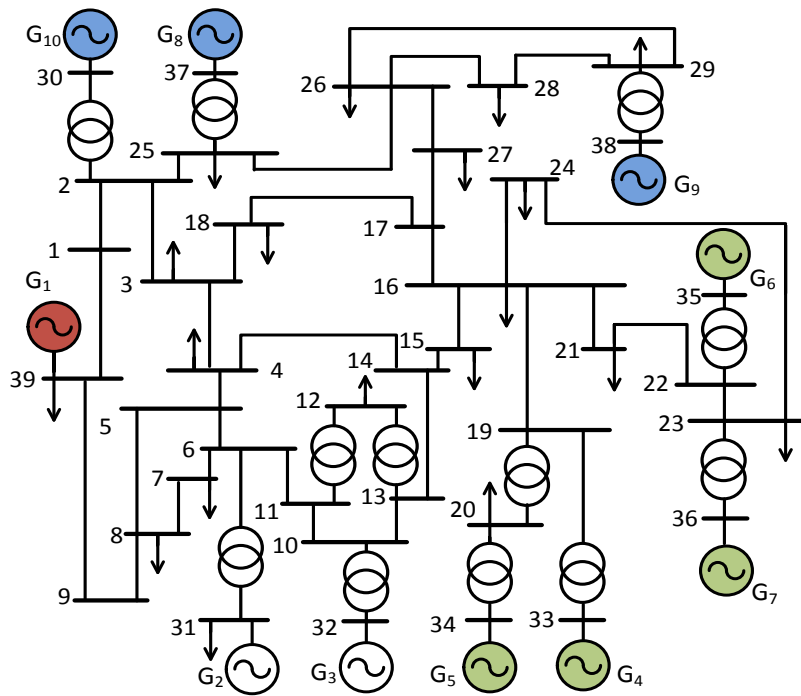


Figure 2.6 IEEE 39-bus New England power system model

can be as small as 0.643 ms when two VMs are in the same location, and as large as 210.774 ms when they are physically far away from each other. To better quantify the impact of delays, in this experiment all the VMs are created in the same site, and controlled artificial latency is injected via (*nsdelay*), similar to the set-up in Section III. It was seen that delay degrades the control performance eventually resulting in instability while the SDN runs under its nominal control mode. The delay optimization controller described in Section III is thereafter deployed. It actively measures the network delay, and adaptively selects the optimal path to minimize the end-to-end delay, maintaining it within the stability bound. In next section, we show the application of the SDN delay optimization controller for wide-area LQR control (8) of the IEEE 39-bus New England power system model (Fig. 2.6).

Table 2.3 The top three paths were calculated where node $N4$ transmits traffic to $N8$. $L1$ and $L8$ are stated UNKNOWN as they represent edge links whose latency cannot be determined as the SDN Controller has no communication to end nodes.

Link	Transmit Latency	Return Latency
L1	UNKNOWN	UNKNOWN
L2	38ms	38ms
L3	32ms	32ms
L4	23ms	23ms
L5	4.1ms	4.1ms
L6	12.6ms	12.6ms
L7	11.8ms	11.8ms
L8	UNKNOWN	UNKNOWN
L3,4,7	66.8ms	66.8ms
L2,7	49.8ms	49.8ms
L3,5,6	48.7ms	48.7ms

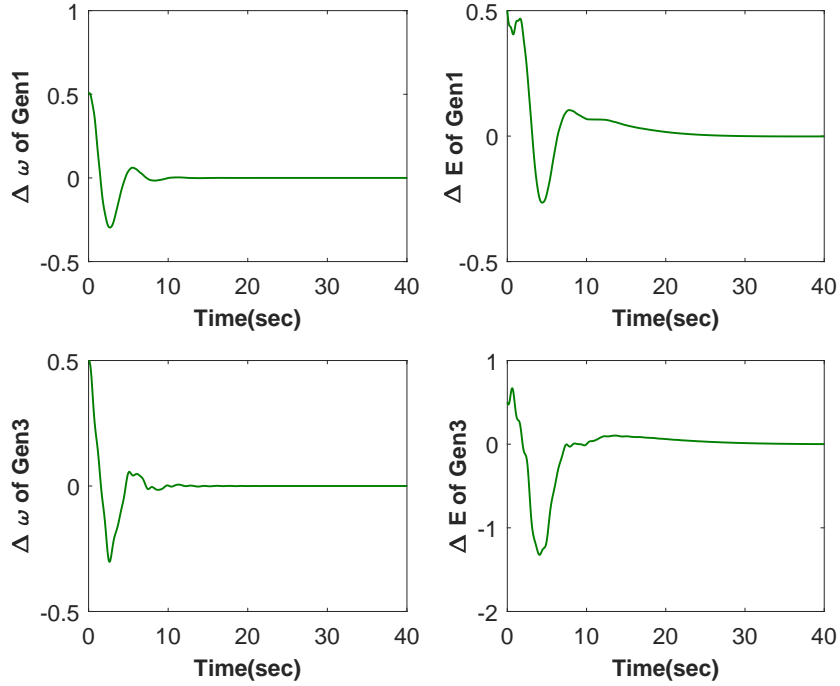


Figure 2.7 Generator state responses under optimized delay control

2.5 Wide-area control in ExoGENI with delay optimization

To show the control performance, we conduct the experiment on the ExoGENI-WAMS testbed described in Section IV. The system model, as described in Section II, is implemented in a master-VM inside ExoGENI, and the control law is implemented using 10 follower VMs. A slice is built in ExoGENI to implement the SDN delay control within the control loop. Figure 2.5 is the topological representation of the experimental wide-area control system. The delay control algorithm is deployed on the "Ctrl" node while the wide-area control algorithm is deployed in nodes $N1$ - $N11$ connected by a SDN network. $N1$ serves as the leader VM, and $N2$ to $N11$ are the local VMs. Similar to the set-up in Section III and IV, *nsdelay* is used to manually inject arbitrary delays into the network links, thereby

emulating the delayed control signal $u = Kx(t - \tau)$ in (2.8).

For testing, *nsdelay* is used to manually inject a 300 ms latency into a pre-selected path, and the controller is tested for selecting a new path after a few seconds of running the iterative control algorithm. Table 2.3 shows the top three paths selected by the SDN controller while node $N4$ communicates to $N8$ when the 300 ms latency is injected. The optimized SDN is seen to select the path that grants a minimum delay, formed by $L3$, $L5$, and $L6$. In other words, it dynamically selects the best path to regulate the delay within the instability threshold (which in this case is 66 ms), thereby guaranteeing closed-loop stability. Figure 2.7 testifies to this stability when the SDN controller selects a new optimal path, and redirects the traffic among the local PDCs. As can be seen from the figure, the generator responses converge to their steady-state values well within 30 seconds.

2.6 Conclusions

This chapter showed an experimental approach for implementing wide-area control using cloud networks. Since the network delay in the shared cloud may exceed the stability tolerance of the grid model, a route optimization method is used for selecting least congested paths in the cloud for PMU data communication. The choice of optimal routes is updated periodically, helping in retaining closed-loop stability. Experimental results show that this framework can effectively address delay stability issues, and improve the closed-loop performance for wide-area control.

CHAPTER

3

ONLINE TUNING OF
CLOUD-BASED WIDE-AREA
CONTROLLERS WITH
VARIATIONS IN NETWORK
TRAFFIC

3.1 Introduction

One lucrative choice for delay-aware wide-area control can be a cloud-based computational framework with SDN as the communication protocol. Similar to any other networks, traffic demands in the cloud will be stochastic and cannot be accurately predicted, implying that network congestion will be unavoidable. Whenever the total input data transmission rate is greater than the output link capacity, congestion will happen. In those situations, the queue length of the measurements may become very large in a short time, resulting in buffer overflow and packet loss, which in turn can not only degrade closed-loop physical response of the grid states, but also threaten stability if the delays exceed a certain upper bound. In chapter 2 [37], a latency control algorithm was proposed for this purpose so that delays in the control loop could be maintained within their stability limit. Although theoretically this framework can let SDNs recompute the shortest path while observing the network behavior between different VMs, in reality the network can still face several challenges. For example, due to the characteristics of the centralized control, it is difficult for a SDN to meet the increasing network demands [26]. The scalability of the controller is yet another challenge for efficient congestion management [46]. Centralized SDN controllers such as *Beacon*, for example, has been reportedly known to face these challenges, while the size, operation, and structure of the controller back-end database (e.g., OpenFlow controller) suffers from tractable controllability. Moreover, the network can also face potential threats from security and dependability [27], such as vulnerabilities in switches, susceptibilities in controllers and lack of mechanisms to ensure trust between the controller and management applications.

Motivated by this problem, in this chapter we propose a dropout mechanism where communication links connecting any pair of generators are dropped when their congestion levels are predicted by the SDN controller to exceed a certain given threshold. Alternatively,

the states can be re-routed through other links, but the mechanism for coordinating that resource allocation may result in an additional delay thereby worsening the end-to-end latency. Our objective, in contrast, is to drop links in a strategic way, and simultaneously tune the control gains associated with the existing links so that the overall closed-loop performance of the wide-area control system remains close to optimal. In recent research, sparsity and optimal control is proposed [43] to reduce the communication costs. For example, sparse linear quadratic regulators (LQR) and H_2 controllers have been proposed by Lin et al. [32] using alternating directions method of multipliers (ADMM), by Wytock and Kolter [60] using proximal Newton method, and by Arastoo et al. [2] using rank-constrained convex optimization, among others. The specific application that we apply this mechanism for is taken as wide-area oscillation damping, which following [17] is posed as a sparse linear quadratic regulator (LQR) design. Decentralized Kalman filters are assumed to be installed at every generator site to estimate the generator states PMU measurements. The LQR gains associated with the exchange of state information between any pair of generators is tuned every time a link is dropped. In other words, we use a sparse wide-area communication graph where the logical structure of the graph is dictated by the time-varying network traffic, while the weights associated with each edge is designed continuously to ensure stability and optimal performance. We illustrate the proposed method using the IEEE 39-bus power system model. Experimental results show that our framework can effectively guarantee the closed-loop performance for wide-area control against the adverse effects of delay regulation failures. How this dropout impact the wide-area controller (2.7), and how stability and close-to-optimal performance can still be maintained after the dropout are presented next.

3.2 Control Objective

We discretize the continuous-time system (2.6) with a zero-order hold and a sampling frequency of $1/T$ (T is the sampling time of PMU), then the discrete-time LTI power system model with a linear state-feedback control can be written as:

$$x(k+1) = A_d x(k) + B_d u(k), \quad (3.1)$$

$$u(k) = K x(k), \quad x(0) = x_0 \quad (3.2)$$

where K is the linear feedback gain for supplementary control, and discrete-time model (3.1) can be excited by the initial state x_0 after a disturbance. It is assumed that the pair (A_d, B_d) is stabilizable.

Our control objective is the optimal distributed control of system (3.1) via a sparse state-feedback control design, in order to reduce the amplitude of oscillations of the closed-loop states. To design a K for (3.2) which satisfies these objectives, we minimize the discrete-time infinite-horizon LQR cost:

$$J_\infty(x_0) = \sum_{k=0}^{\infty} \{x(k)' Q x(k) + u(k)' R u(k)\} \quad (3.3)$$

under the structural constraint $K \in \Omega$, where Ω is the set of matrices with pre-specified zero locations, and hence enforces the desired sparsity structure. $Q > 0$, $R > 0$ are the state and control weighting matrices respectively, where Q is designed such that the square of the differences of the linearized generator angles (proportional to the power transfer between generators), along with the quadratic energy of the rest of the states are penalized. This

translates to the state-weighting term in (3.3) of the form:

$$x(k)'Qx(k) = \sum_{i=1}^n \sum_{j>i}^n (\Delta\delta_i(k) - \Delta\delta_j(k))^2 + \sum_{i=1}^n \tilde{x}(k)' \tilde{x}(k) \quad (3.4)$$

where \tilde{x} is the vector of all linearized states except the rotor angle state $\Delta\delta$. The feedback gain matrix K in (3.2) can be designed by a central control authority, which then communicates its individual rows $K(i, :), \forall i = 1, \dots, n$, to the corresponding generator actuators, i.e. $u_i(t) = K(i, :)x(t); \forall i = 1, \dots, n$. Then, depending on the non-zero entries of $K(i, :)$, the i^{th} controller can communicate and receive the corresponding states from other generators to calculate its own actuator control input. Hence it is clear that if sufficient number of the K matrix entries are zero, the communication graph will be sparse, and the control architecture will fall under the category of distributed control. The following section provides modal analysis of the system, which forms a basis for constructing the structure of K .

3.3 Structurally Constrained Optimal Control

3.3.1 Construction of Feedback Structure

Following the strategy proposed in the previous section, for the control signal vector $u = Kx$, the state feedback matrix is following with the structure matrix Ω . The procedure for constructing Ω is generalized as follows.

Let the link connecting VM i to VM j be denoted as ℓ_{ij} . If ℓ_{ij} is dropped because of congestion, then state $x_i(t)$ can no longer be transmitted to VM j for computing u_j . One

way to model the impact of this is to define a binary matrix Ω where:

$$\Omega(i, j) = \begin{cases} 0 & \text{if delay regulation fail} \\ 1 & \text{else.} \end{cases} \quad (3.5)$$

For example, consider a 4-machine power system, where 4 VMs are constructed in the cloud. We assume the self-delays to be always zero, meaning the state arriving at VM i can be used for computing u_i at that VM without any delay. Table 1 shows all possible dropout scenarios in the 16 uni-directional links for this 4-VM cloud. Thus, the control law $u = Kx$

Table 3.1 Construction of Ω for a 4-VM example

From VM i to VM j	link ℓ_{ij}	fail ?	$\Omega(i, j)$
1 to 1	/	/	1
1 to 2	1	Yes	0
1 to 3	2	No	1
1 to 4	3	Yes	0
2 to 1	4	Yes	0
2 to 2	/	/	1
2 to 3	5	No	1
2 to 4	6	No	1
3 to 1	7	No	1
3 to 2	8	No	1
3 to 3	/	/	1
3 to 4	9	No	1
4 to 1	10	Yes	0
4 to 2	11	No	1
4 to 3	12	No	1
4 to 4	/	/	1

for this example can be constructed as

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}}_{\Omega} \circ K \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad (3.6)$$

where \circ denotes Hadamard product, and $\mathbf{1}$ and $\mathbf{0}$ are row vectors of 1s and 0s respectively for block sparsity. The matrix K , however, should not be the same as the LQR gain matrix, as there is no guarantee that its Hadamard product with Ω will be stabilizing, or will guarantee satisfactory closed-loop performance. With each dropout, i.e. with the construction of a new Ω every time, K must be simultaneously re-computed to ensure that the closed-loop is stable and close to optimal. Next, we formulate the the optimal control design procedure using LQR control, with the state-feedback structure constructed above as a constraint.

3.3.2 Discrete Linear Quadratic Regulator

A constrained optimal control formulation for the power system is done by specifying structural constraints on the feedback matrix, and then using a generalized Riccati equation (GDARE) method to obtain a stabilizing constrained K . The infinite-horizon quadratic cost (3.3) to be minimized is re-written here for continuity as follows:

$$J_{\infty}(x_0) = \sum_{k=0}^{\infty} \{x(k)'Qx(k) + u(k)'Ru(k)\} \quad (3.7)$$

The linear state-feedback control law is given by $u(k) = K x(k), \forall k > 0$, and the optimal K is obtained from:

$$K = -(R + B_d' P B_d)^{-1} B_d' P A_d \quad (3.8)$$

where the unique symmetric matrix $P > 0$ is obtained from the well-known discrete algebraic Riccati equation (DARE):

$$P = A_d' P A_d - A_d' P B_d (R + B_d' P B_d)^{-1} B_d' P A_d + Q \quad (3.9)$$

where the optimal cost can then be given by $J_\infty^* = x_0' P x_0$. In addition to the dynamic system constraint (3.2), we add the structural sparsity constraint $K \in \Omega$, where Ω is determined from network condition in the above section. The constrained control problem then becomes:

$$\mathbb{P}: \quad \min J_\infty(x_0), \quad (3.10)$$

$$\text{s.t.}, K \in \Omega. \quad (3.11)$$

Also, the solution of \mathbb{P} should provide a matrix K such that the closed-loop system matrix $A + BK$ is Hurwitz. It is clear that the nominal solution from (3.8) will no longer necessarily satisfy the structural constraint (3.11). In the next subsection, we propose an online constrained control design which provides a solution to the above problem.

3.3.3 Generalized Riccati Equation Method

Following [24], let I_Ω be the indicator matrix for the structured matrix Ω , and I_Ω^c be the complement of I_Ω . Then, as shown in [24], it can be easily verified that the following identity

holds for the structural constraint in (3.11):

$$F(K) \triangleq K \circ I_{\Omega}^c = \mathbf{0} \quad (3.12)$$

where \circ is the Hadamard product, and $\mathbf{0}$ is the zero matrix of appropriate dimension. For the discrete-time system and control law, the state-feedback gain matrix K satisfies

$$K + L = -(R + B_d' P B_d)^{-1} B_d' P A_d \quad (3.13)$$

where $'$ denotes matrix transpose, and A_d and B_d are the discrete-time state and input matrices of the power system model. The matrix $P = P' > 0$ is the solution of the Generalized Discrete Algebraic Riccati Equation (GDARE) [24]

$$P = A_d' P A_d - A_d' P B_d (R + B_d' P B_d)^{-1} B_d' P A_d + Q + L' (R + B_d' P B_d) L. \quad (3.14)$$

If there exists a solution P for the above equation for the sparsity-dependent matrix L , then the closed-loop system is guaranteed to be stable [24]. The matrix L for our purpose can be chosen as

$$L = F(\Psi(P)) \quad (3.15)$$

where: $\Psi(P) \triangleq -(R + B' P B)^{-1} B' P A$.

3.3.4 Algorithm Solution of \mathbb{P}

The procedure of solving the GDARE (3.14), and checking the local convergence of the normalized P iteratively can be carried out by a numerical algorithm, referred to as Geromel's algorithm. The steps are shown in Algorithm 1. Once the algorithm converges for a small

value of ε , the sparse feedback matrix K is calculated from (3.13).

Algorithm 1 Algorithm for solving \mathbb{P}

- 1: Initialization: Compute P^0 by solving the DARE from the well-know DARE: $P = A'PA - A'PB(R + B'PB)^{-1}B'PA + Q$
 - 2: Start from $k = 0$,
 - 3: Compute $L^{k+1} = \Psi(P^k) \circ I_{\Omega_i}^c$, in which, $\Psi(P^k) = -(R + B'P^k B)^{-1}B'P^k A$
 - 4: $Q^{k+1} = Q + (L^{k+1})'(R + B'P^k B)L^{k+1}$
 - 5: Calculate P^{k+1} by solving the GDARE: $P^{k+1} = A'P^{k+1}A + A'P^{k+1}B\Psi(P^{k+1}) + Q + (L^{k+1})'(R + B'P^k B)L^{k+1}$
 - 6: Check convergence:
 If $err(k) = \frac{\|P^{k+1} - P^k\|_2}{\|P^0\|_2} < \varepsilon$, $K = \Psi(P^k) - L^{k+1}$;
 else $k = k + 1$ and go to Step 3.
-

3.4 Implementation on ExoGENI-WAMS Cloud Testbed

The implementation of the sparse wide-area controller is demonstrated using the ExoGENI-WAMS testbed [18]. This testbed consists of two parts. The physical layer consists of the power system model implemented in real-time digital simulators (RTDS) and hardware PMUs generating the state measurements at every generator. The cyber layer is represented by the ExoGENI distributed cloud environment which connects to the physical system via high-bandwidth virtual network links. ExoGENI has multiple cloud sites geographically distributed over the world that can provide on-demand virtual topologies of different scales and capabilities. A virtual topology is created with customized VMs interconnected by virtual network links. ExoGENI provides both graphical interface (ORCA-Flukes) and programmable APIs for creating and managing this virtual topology by which wide-area communication can be emulated in real-time. Fig. 3.1 illustrates this cyber-physical ar-

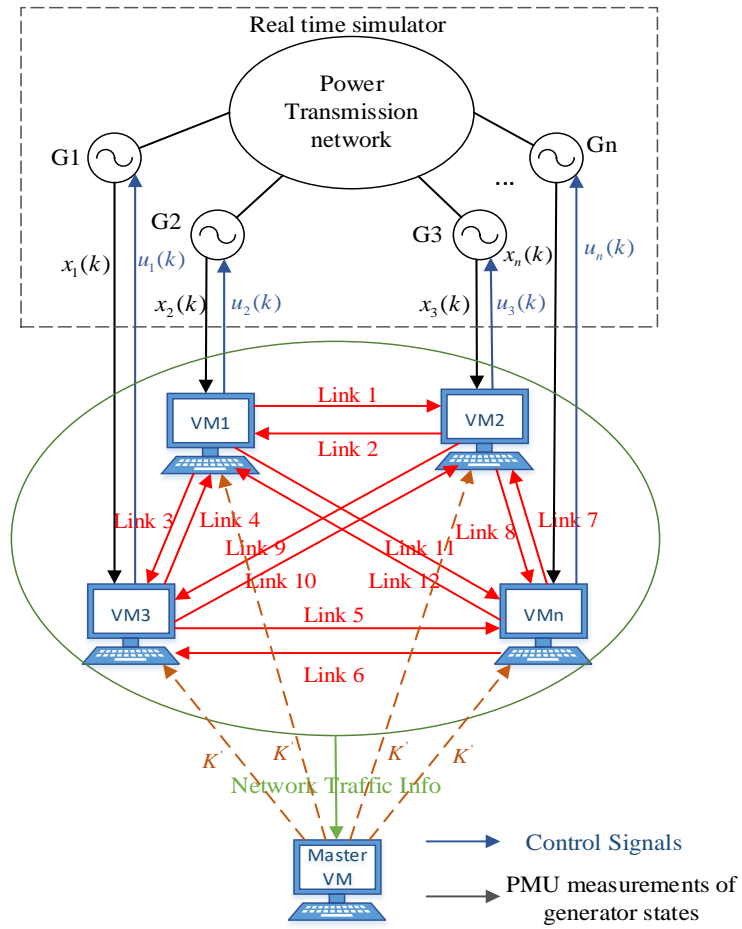


Figure 3.1 Architecture of the ExoGENI-WAMS Testbed

chitecture that implements the proposed cloud-in-the-loop wide-area controller (2.7). In this figure, PMUs are installed at every generator bus. The PMU data from generators in a specific area in the grid are aggregated at a local control center, denoted as a local PDC. The data are then communicated to the control application hosted in the Cloud. Each generator is only allowed to communicate with its own local control server implemented in a VM in its designated Cloud site. For the sake of the experiments here, a leader VM is used to generate the PMU data and receive control signals. This leader VM generates the measurement signals at every sample time and broadcasts them to all the local VMs. Upon receiving the data, the local VMs exchange the information they received, compute the control signals and send back to leader VM, awaiting measurements for the next sampling instant. The communications between the local VMs will incur large random delays as they are distributed over a wide-area network, decided collectively by the VM topology, physical locations of VMs, and the background network traffic. A master VM is used to periodically monitor this network traffic, to make decisions of dropping congested links, and accordingly to generate the state feedback structure Ω (3.5)-(3.6) using which it updates the feedback control gains based on Algorithm 1.

3.5 Simulation Results

To verify the performance of proposed method, the IEEE 39-bus model of New England power system is used. This model has 10 generators, with a total of 130 states, including the exciter and the PSS. We conduct the experiment on the ExoGENI-WAMS testbed described in Section V. The system model, as described in Section II, is implemented in a *leader* VM, and the control law is implemented using 10 follower VMs. A slice is built in ExoGENI to implement the SDN delay control within the control loop. Subsequently, a master VM is used

to construct the sparse structure Ω and compute the sparse feedback K and then broadcast the new block-row K_i of the control gain to the corresponding VMs. To quantify the impact of sparse control algorithm, delay are manually injected. These delay are drawn from Gaussian distribution for the simulations. Following the approach described in Section III, the sparsity structure is constructed periodically once SDN delay regulation fails. This failure is emulated artificially for the sake of the simulations by creating artificial congestion in designated sets of links, randomly over time. Algorithm 1 is implemented to tune the corresponding block-sparse feedback matrices. The convergence parameter is set as $\varepsilon = 10^{-3}$.

We present simulations results starting from the ideal all-to-all connected network for the LQR wide-area controller, and progressively dropping links as the traffic in specified links are observed or predicted to be high. The threshold for link dropout is set as $h = 5T$, meaning if any state vector is delayed by more than five sampling intervals the corresponding link is considered to be congested. The performance of the controllers for different traffic status in terms of the value of the closed-loop objective function (3.7) are reported in Table 3.2. Fig. 3.2 illustrates the increasingly sparse network topology as traffic worsens in the links. For simplicity, self-links are not shown in Fig. 3.2. The traffic status of the links is shown in Table 3.2. Fig. 3.3 shows the corresponding closed-loop responses of the ten generator frequencies. As expected, the value of J increases as the sparsity level increases, i.e., as more links are dropped. But because of the re-tuning via Algorithm 1, the closed-loop response of the frequencies still remain close to the optimal LQR response, except for small deviations as expected from the sparsity. The optimally tuned sparse controller at the end of all considered cases is observed to provide an acceptable control performance with as high as 80% link loss. Fig. 3.4 shows that if these links are dropped but K is not tuned, then the closed-loop system becomes unstable. This validates the use of Algorithm 1.

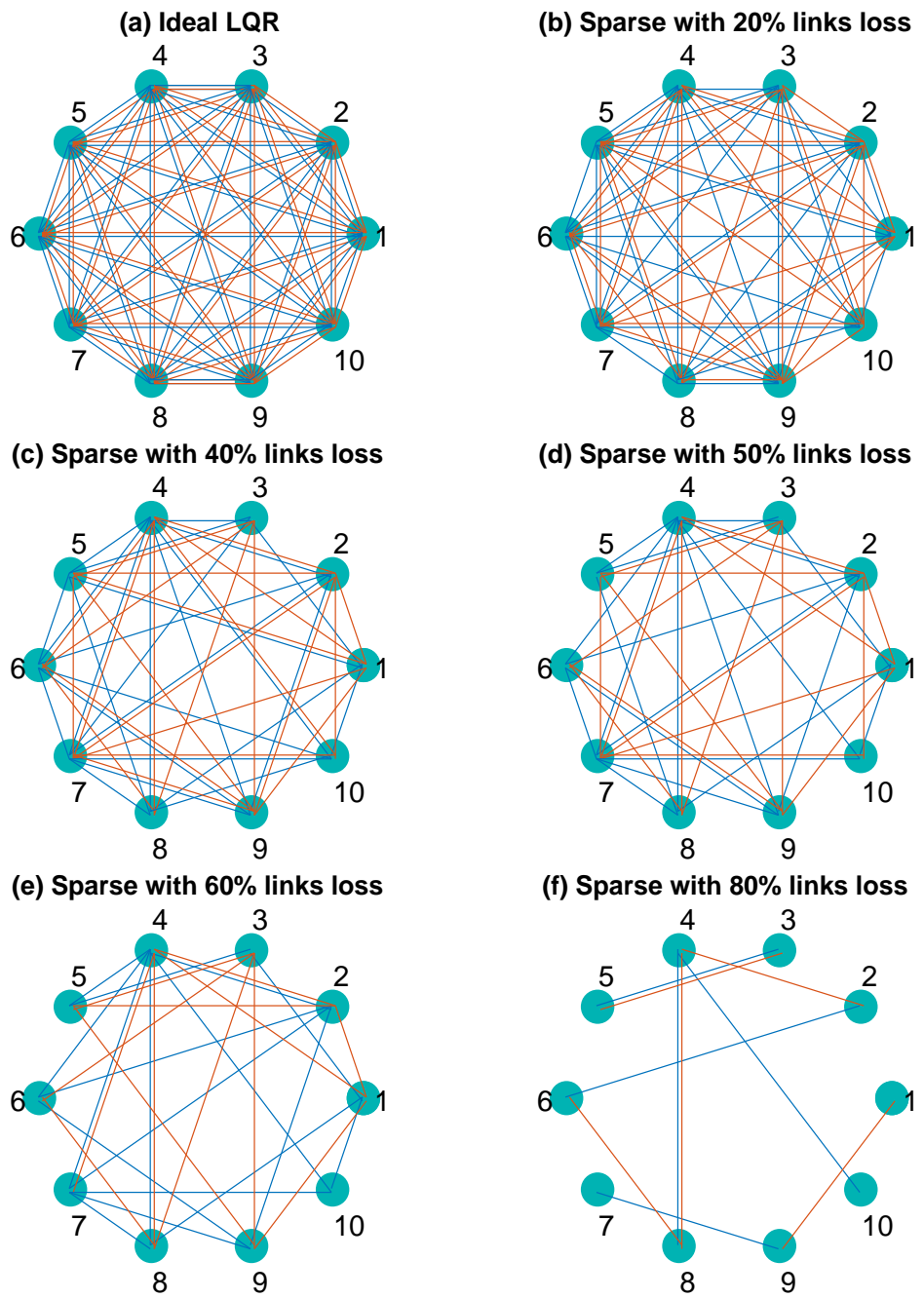


Figure 3.2 Network topology of different situations

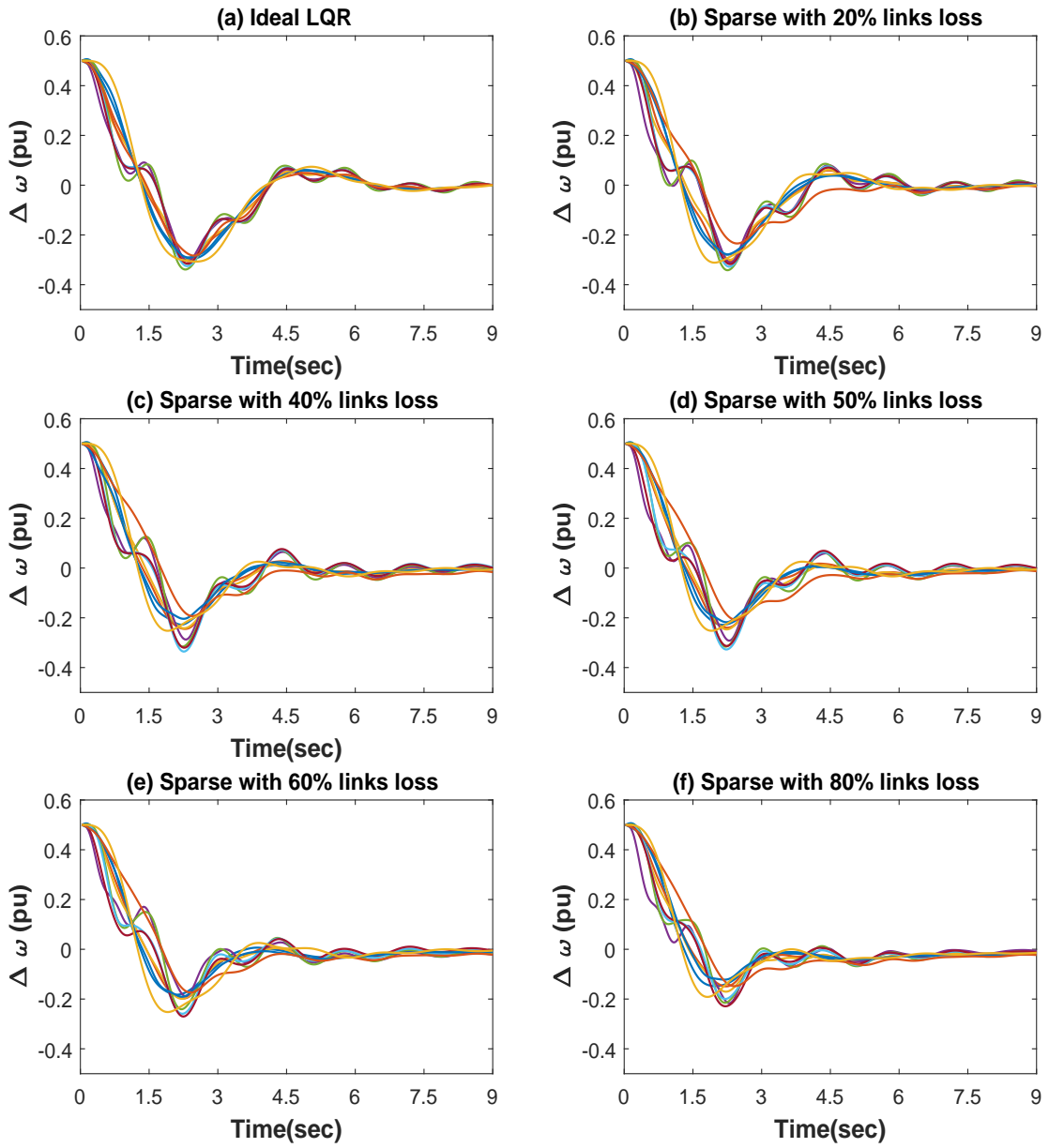


Figure 3.3 Performance of different controllers in damping oscillations

Table 3.2 Network status and Value of J

Time instant	Traffic status	Values of J
$t = t_0$	$\tau_{i,j} < \tau_{th}$, for all(i,j)	2.3156
$t = t_1$	$\tau_{i,j} < \tau_{th}$, for 20% of links	2.9518
$t = t_2$	$\tau_{i,j} > \tau_{th}$, for 40% of links	3.5521
$t = t_3$	$\tau_{i,j} > \tau_{th}$, for 50% of links	3.9046
$t = t_4$	$\tau_{i,j} > \tau_{th}$, for 60% of links	4.4287
$t = t_5$	$\tau_{i,j} > \tau_{th}$, for 80% of links	5.6739

3.6 Conclusions

In this chapter, we presented a wide-area controller tuning algorithm based on a sparse communication graph determined by time-varying network traffic. A constrained linear quadratic regulator (LQR) is designed to balance the trade-off between the control performance and the level of sparsity resulting from link dropouts. Experimental results show that our framework can effectively guarantee the closed-loop performance for wide-area damping control even when the network controllers fail to regulate delays within desired bounds.

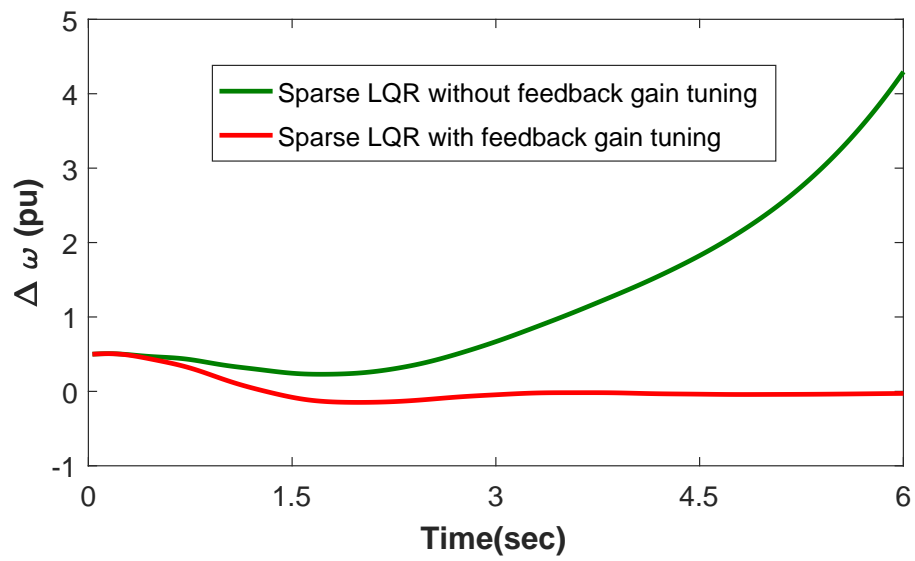


Figure 3.4 Performance of sparse control algorithm

CHAPTER

4

**ONLINE ADAPTIVE WIDE-AREA
DAMPING CONTROLLER BASED
ON DYNAMIC DELAY AREA
DETECTION USING KERNEL
DENSITY ESTIMATION**

4.1 Introduction

When communication occurs over shared resources, delays often arise that may have destabilizing effects on the closed-loop system. In order to ensure an optimal control design under such delays, a delay-aware controller such as sparse control is essential. However, the network condition is dynamic in real applications. The delays may appear occasionally and network can get recovered even when encounters obvious delays. If the network recovers but the controller used is still designed for delayed network, then the control performance will be spoiled.

In chapter 3 [36], a dropout mechanism was presented where communication links connecting any pair of generators are dropped when their congestion levels are measured by the SDN controller to exceed a certain given threshold. However, the more links dropped, the worse the control performance is. In practical situation, the network delay is not constant but time-varying, therefore, it is obviously not optimal if keep dropping links but never bringing back the dropped links even if the network recovers, where the control performance is limited. Thus, to optimize the control strategy based on network condition, in this chapter, we first point out and prove the time series forecast model is inadvisable in delay prediction, and then propose a kernel density estimation (KDE) based model to dynamically detect adverse network delay area by monitoring the round trip delay of interconnected links. Once a link is detected entering or out the delay area, we simultaneously tune the control gains associated with the existing links so that the overall closed-loop performance of the wide-area control system remains approximated optimal. We validate the effectiveness of our proposed method using the real network data collected from ExoGENI cloud platform and simulations of the IEEE 39-bus power system model. Experimental results show that our framework can effectively guarantee the closed-loop performance for

wide-area control in the time-varying network traffics.

4.2 Online Adverse Delay Area Detection via Machine Learning Method

Ideally, to achieve the best online control performance, the ultimate optimal control system should act accordingly at each time stamp for each single delay point, which can be easily measured by the network round trip time (RTTs). However, unfortunately, it is costly and unrealistic to change control law as soon as abnormal network point detected at each time stamp, especially considering the robust of control system. Thus, we need to recognize the "true" delay area which will have real negative effect for control performance considering the inherent control robust ability.

The online networking data is actually a time series. However, since the network delay is a highly random process and also is known as a non-stationary sequence which is very hard to be transformed to stationary time series. In this section, we first used a real example to illustrate why traditional time series forecasting method, i.e., Autoregressive Integrated Moving Average Model (ARIMA) is invalid here and then introduced our effective Kernel Density Estimation (KDE) based delay detection method.

4.2.1 Data Collection

To monitor the network condition for our online adaptive control, the Internet path between RENC1 (Chapel Hill, NC USA) and OSF (Oakland, CA USA) was continuously observed for data collection. Packet delay data were collected as a discrete-time series using the ping tool under ExoGENI platform. RTTs were measured by sending ICMP (Internet Control

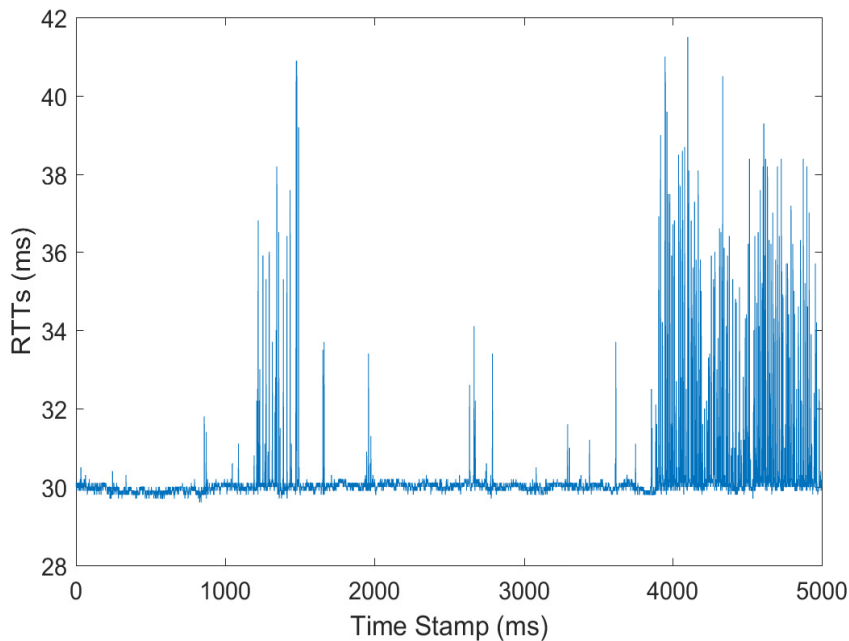


Figure 4.1 Example of RTT variations between RENCi (Chapel Hill, NC USA) and OSF (Oak-land, CA USA) on ExoGENI cloud platform

Message Protocol, RFC 792) echo-request to a target machine and responds with echo-reply messages. Without loss of generality, we set the ICMP echo packet as 64 Bytes for simulation purpose and the size can be adjusted for better network monitoring in real application. Figure 4.1 is an example of 5000 observed RTTs between RENCi and OSF, and we will use this dataset for the following use. We conducted the experiments based on these data collected.

4.2.2 Invalid Time Series Forecasting Model for Delay Prediction

As the data we collected is a time series, we naturally tried to use time series forecasting model to do the delay prediction for our further control purpose. ARIMA, one of the most famous time series forecasting model, was applied to the data collected above. We used first 250 data to train the model and the rest data is for testing. Rolling horizon strategy is

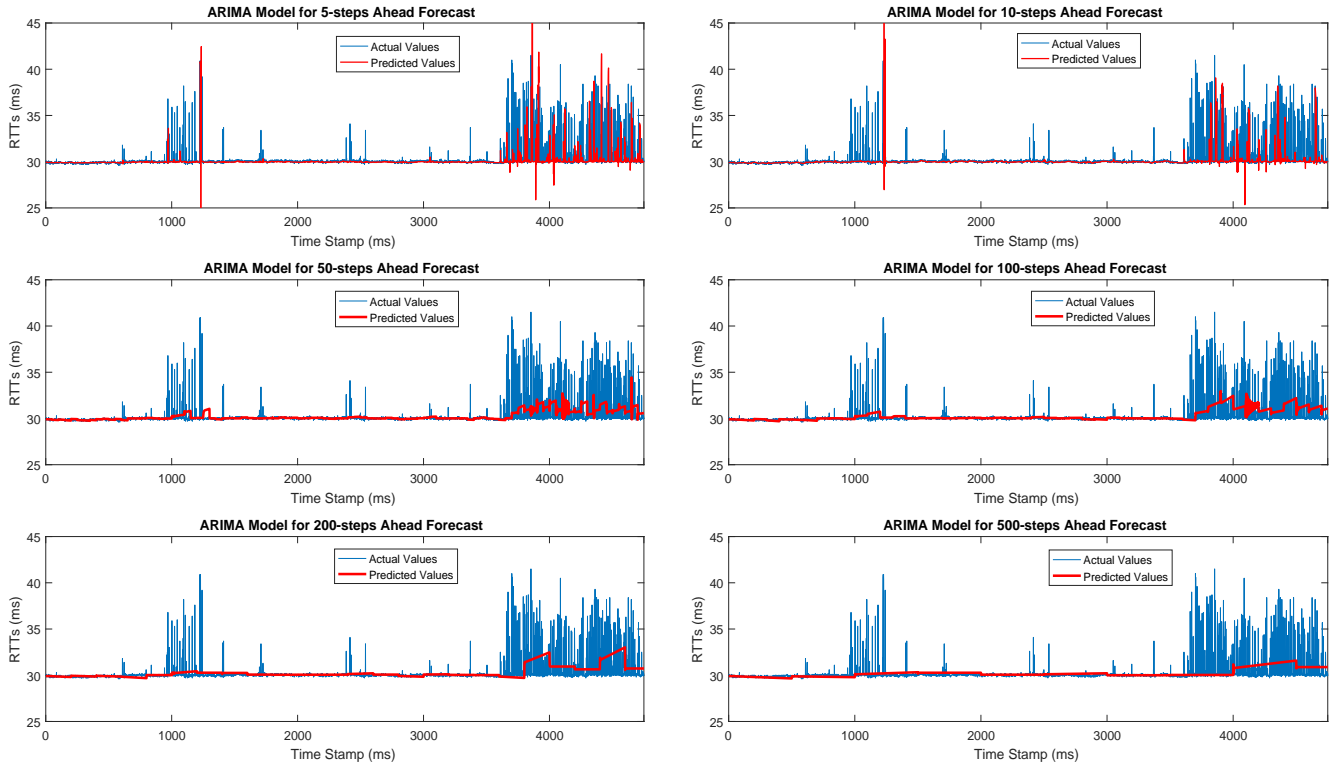


Figure 4.2 Time series of the RTT observations and the forecasts provided by ARIMA model for different steps ahead

used here which means, the passed test data can be used for training the updated model. We conducted different steps ahead forecasting experiments and reported the overall performance in Figure 4.2 and the Root Mean Square Error (RMSE) results in Table 4.1. As can be seen from Figure 4.2, the forecasting performance of each model is not satisfied, since the best forecasting model (observed from RMSE value), 50-steps ahead, is still far from being used for control purpose. Thus, we need to explore other effective ways to detect delay area.

Since the network delay is difficult to predict, we propose to use machine learning models to do the delay pattern recognition based on the existing RTTs data. In this way, ideally, we can know when the potential major network delay is detected and when the network is

Table 4.1 The comparison of RMSE for different ARIMA forecasting steps

Forecasting steps	RMSE value
5	2.5375
10	1.9091
50	1.2280
100	1.2419
200	1.3397
500	1.2571

recovered so the adverse delay area can be determined.

In statistics, Kernel density estimation (KDE) is non-parametric way to estimate the probability density function of a random variable, which is very suitable for patten recognition here. Let (x_1, x_2, \dots, x_n) be a univariate independent and identically distributed sample from an unknown distribution P with density function p . KDE can be expressed as

$$\hat{p}_n(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right) \quad (4.1)$$

where K is a smooth function called the kernel function and $h > 0$ is the smoothing bandwidth that controls the amount of smoothing. We assume if the network encounters a major delay, the delay is likely continued for a certain period time. Thus, we use Gaussian kernel as $K(x)$ to better estimate the continues density, where $K(x) = \frac{e^{(-\|x\|^2/2)}}{v_{1,d}}$, $v_{1,d} = \int e^{(-\|x\|^2/2)} dx$. As we know, it is not difficult to identify if a certain detected RTT is abnormal (delayed) or normal (non-delayed). This can be simply achieved by either setting a threshold (e.g., PMU sample time) or using other machine learning models (e.g., K-Nearest Neighbor) to do the anomaly detection. Here for simplicity, we use PMU sample time (PST) as example to illustrate our proposed KDE based adverse delay area detection method, as shown in Algorithm 1. The parameters are defined as below. 1) Window size (WS): The most WS recent RTTs

taken into consideration; 2) Density threshold (DT): The threshold of calculated density for each point in the window; 3) The number of star points (NS): the number of point with density value greater than DT ; 4) The length of continuous NS (CNS); 5) The local search step (LSS): the local search step when the abnormal point is detected; 6) The local search threshold ratio ($LSTR$): The ratio based on average RTTs in LSS for calculating abnormal threshold for local search.

Algorithm 2 Algorithm for KDE based adverse delay area detection

- 1: Initialization: Set all the parameters
 - 2: For each RTT in WS , If $RTT > PST$, $t_{status} \leftarrow$ abnormal; Else $t_{status} \leftarrow$ normal
 - 3: If t_{status} is abnormal, do the local search:
 - 4: For each RTT in LSS , if $RTT - avg_{RTTs} > (PST - avg_{RTTs}) * LSTR$, $t_{status} \leftarrow$ abnormal
 - 5: Calculate density p based on each t_{status} in WS . Then calculate NS and CNS based on p .
 - 6: If $NS > NS_0$ and $CNS = CNS_0$, then "Delay Detected"
 - 7: Else if the length of continuous Non-Star points = CNS_0 , then "Network Recovered".
-

Figure 4.4 shows the results of the proposed model with the parameter $WS = 100$, $DT = 0.1$, $LSS = NS = 5$, $CNS = 3$ and $LSTR = 0.5$. We can our model can accurately identify the delay area which may have adverse effect for the control performance. Actually, we can tune the model sensitivity to identify the start and end of delay area by setting different parameters above. For example, if we set a bigger LSS , the model will be more "cautious", which means it responses quickly for each potential delay area. But the side-effect is it may frequently switch from the delay and normal network situation which may increase the burden of computing resources of the network. Thus, users can adjust those

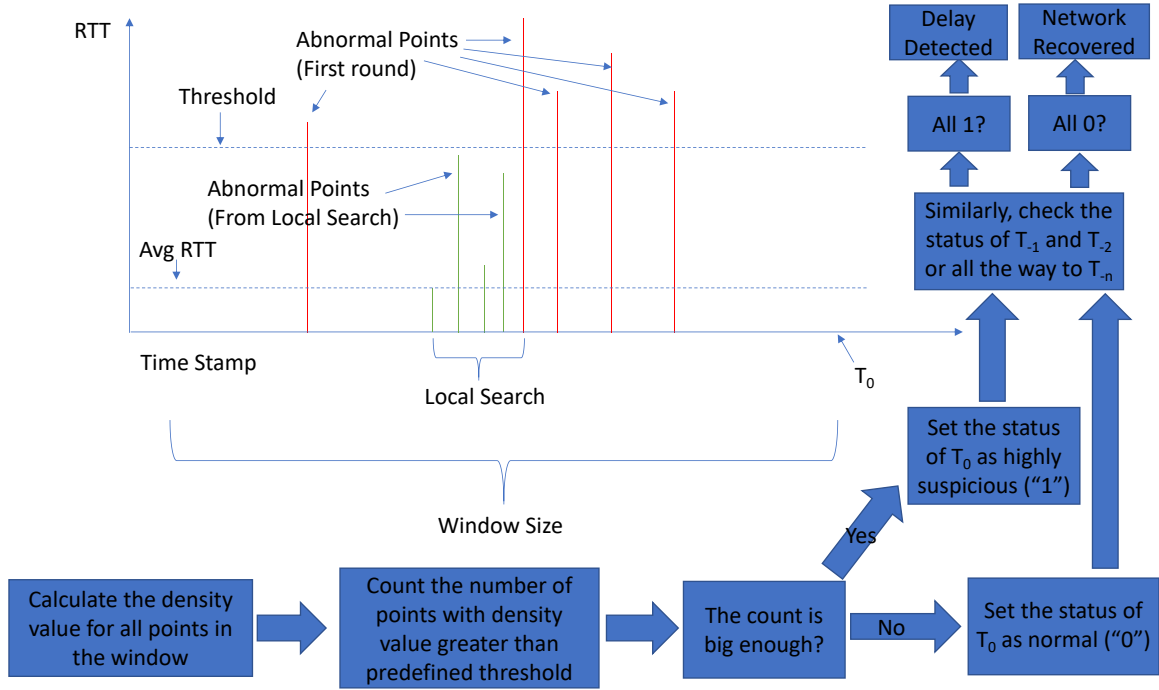


Figure 4.3 Flowchart for Delay Area Detection algorithm by KDE-based Model

parameters in real practice based on their preferences. In the following, a sparse feedback control for designing real-time controller is illustrated once the delay area is detected by KDE.

4.3 Construction of Sparse Control Structure

Let the link connecting VM i to VM j be denoted as l_{ij} . If l_{ij} is dropped because of delay, then state $x_i(t)$ can no longer be transmitted to VM j for computing u_j . One way to model the impact of this is to define a binary matrix Ω where:

$$\Omega(i, j) = \begin{cases} 0 & \text{if delay detected} \\ 1 & \text{else.} \end{cases} \quad (4.2)$$

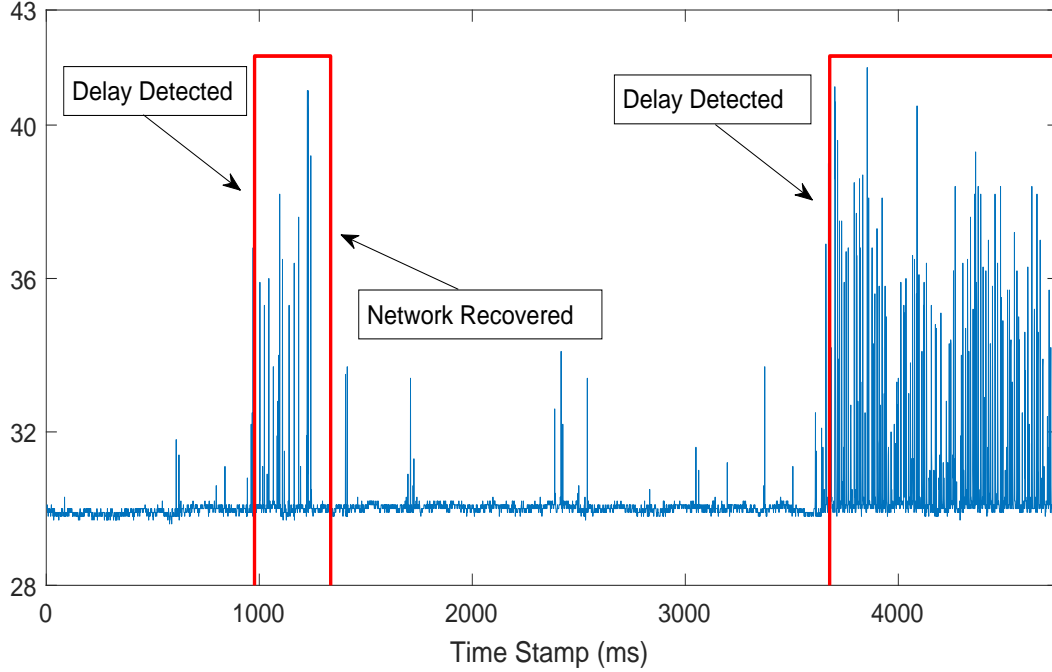


Figure 4.4 Adverse Delay Area Detected by the Proposed KDE-based Model

For example, consider a 3-machine power system, where 3 VMs are constructed in the cloud. We assume the self-delays to be always zero, meaning the state arriving at VM i can be used for computing u_i at that VM without any delay. Figure 4.5 shows an example of 3 bi-directional links network conditions for this 3-VM cloud. Here "1" indicates delay conditions while "0" represents no delay conditions. Line1 is the link between node1 and node2, line2 is the link between node2 and node3, and line3 is the link between node1 and node3.

$$t_1 - t_2 : \Omega = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

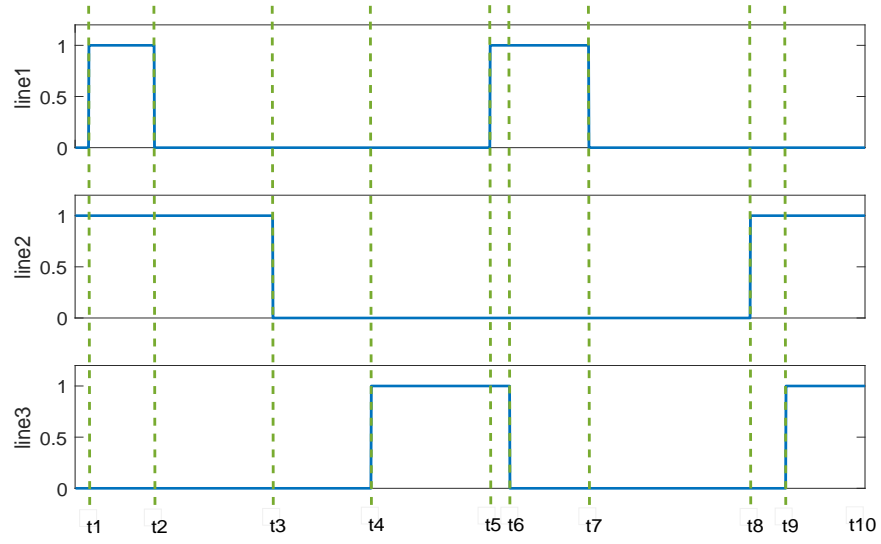


Figure 4.5 Example of network conditions for 3 bi-directional links

$$t_4 - t_5 : \Omega = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$t_6 - t_7 : \Omega = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$t_9 - t_{10} : \Omega = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus, the control law $u = Kx$ for this example from t_1 to t_2 can be constructed as

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}}_{\Omega} \circ K \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad (4.3)$$

where \circ denotes Hadamard product, and $\mathbf{1}$ and $\mathbf{0}$ are row vectors of 1s and 0s respectively for block sparsity. The matrix K , however, should not be the same as the LQR gain matrix, as there is no guarantee that its Hadamard product with Ω will be stabilizing, or will guarantee satisfactory closed-loop performance. With each delay detected or link recovery, i.e. with the construction of a new Ω every time, K must be simultaneously re-computed to ensure that the closed-loop is stable and close to optimal. One mechanism to ensure this is described next.

4.4 Simulation Results

The IEEE 39-bus model of New England power system is used to verify the performance of proposed method. This model has 10 generators, with a total of 130 states, including the exciter and the PSS. A sampling time of 1/30 secs is selected for discretizing the model. The system model, as described in Section II, is implemented in a *leader* VM, and the control law is implemented using 10 follower VMs. A slice is built in ExoGENI to monitor the real time delay and detect the delay area of the interconnected links via the proposed KDE based model. The parameters are the same as in Section III. Subsequently, a master VM is used to construct the sparse structure Ω and compute the sparse feedback K and then broadcast the new block-row K_i of the control gain to the corresponding VMs. Following

the approach described in Section III, the sparsity structure is constructed dynamically once adverse delay detected or network recovered. Algorithm 2 is implemented to tune the corresponding block-sparse feedback matrices. The convergence parameter is set as $\varepsilon = 10^{-3}$. To better compare the control performance under dynamic network conditions, starting the simulations at $t = 0$, we introduce two disturbances at $t = 10$ secs and $t = 37$ secs respectively.

We present simulations results starting from the ideal all-to-all connected network for the LQR wide-area controller, and progressively dropping links or bringing back links as the adverse delay or network recovery in specified links are detected by KDE. Three different cases shown in Table 4.2 are tested to validate the control performance. Fig. 4.6 shows the

Table 4.2 3 Cases for validating the control performance

Cases	Delay	KDE	Sparse Algorithm
Case 1	No	No	No
Case 2	Yes	No	Yes
Case 3 (Proposed)	Yes	Yes	Yes

corresponding closed-loop responses of generator 2. Case 1 is an ideal condition in which the network are fully linked and dense LQR feedback control is applied. Case 2 indicates that the system can keep stable when sparse algorithm is used even if the network suffers from delay. Compared with case2, our proposed framework (Case3) performs much better as our model can dynamically tune the feedback gains based on different network conditions. Furthermore, considering the proposed framework can satisfy various control demands by setting different model parameters based on user's preference, it is very promising in real applications.

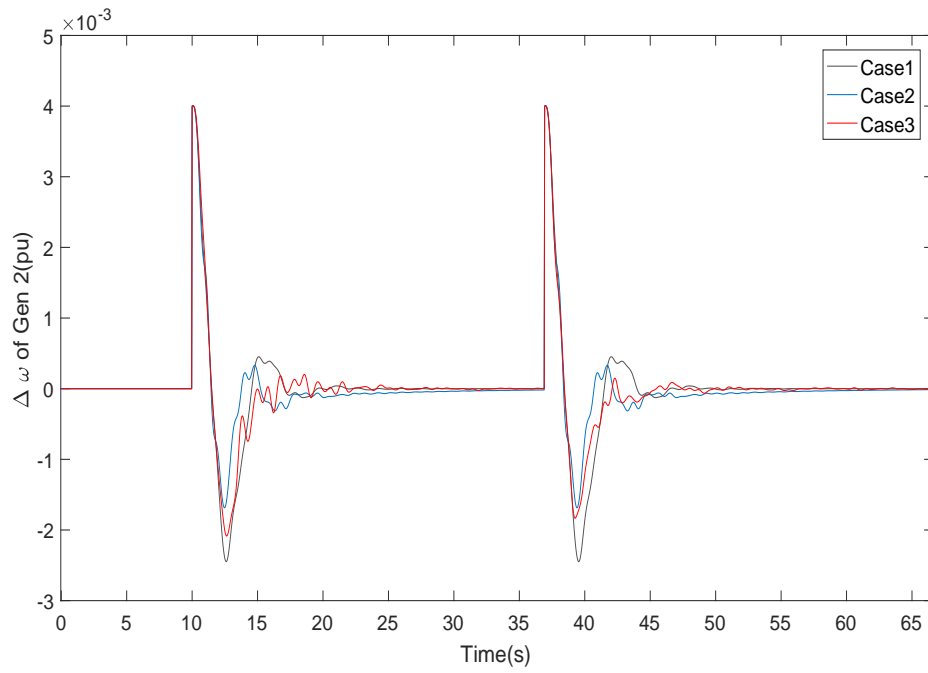


Figure 4.6 Performance of different cases in damping oscillations

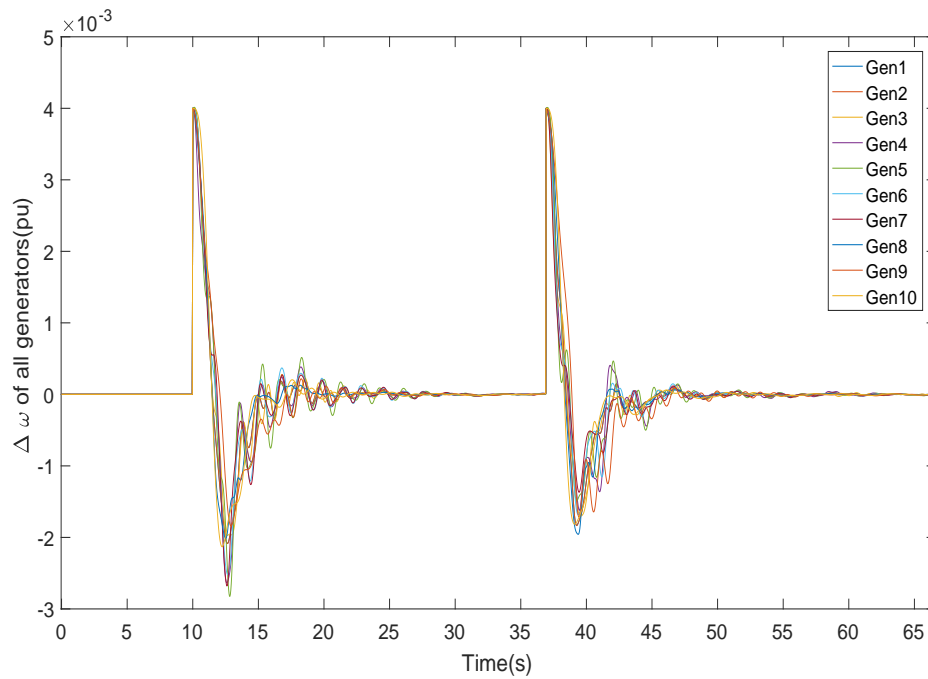


Figure 4.7 Output response for all generators

4.5 Conclusions.

In this chapter, we first presented a KDE based model to identify the adverse delay area which may potentially impact the control performance. Then a sparse communication graph is adjusted according to the detection results and a sparsity-constrained LQR algorithm is dynamically applied, to achieve the optimal control performance. Experimental results show that our proposed algorithm can effectively guarantee the closed-loop performance for wide-area damping control under various network conditions.

CHAPTER

5

DELAY AWARE DISTRIBUTED
DATA-DRIVEN CONTROL
METHOD FOR WIDE-AREA
POWER SYSTEMS

5.1 Introduction

In chapter 3, a dropout mechanism was presented where communication links connecting any pair of generators are congested when their congestion levels are measured by the SDN controller to exceed a certain given threshold. However, the more links dropped, the worse the control performance is, and also the more links dropped, the more time need to be used to compute a sparse feedback control gains. In practical situation, the network delay is not constant but time-varying, therefore, it is obviously not optimal if keep dropping links but never bringing back the dropped links even if the network recovers, where the control performance is limited. Furthermore, the control law needs to be modified promptly as the congested link changed shown in chapter 4, which requires a high performance computing capacity for the generator connected VMs. Besides network delay, packet loss is also a significant issue that cannot be ignored during the data transmission. Conventionally, researchers use the PMU data to identify the physical process[12] while in this chapter we will focus recover the missing PMU data. Gao et al.[14], [20] developed techniques to recovery missing PMU data caused by network delay. The results mainly utilize the low-rank property of synchrophasors for data recovery. However, such low-rank-based methods cannot properly handle missing data caused by communication latencies since the available data may not satisfy the minimal obtained measurement requirement. In addition, the employed synchrophasor completion methods involve data approximation, which can potentially undermine the data recovery accuracy.

To address this issue, in this chapter, we propose a delay aware prediction control (DAPC) framework to address the data transmission of PMU measurements in modern power system. DAPC is designed to complete the current states and predict the future states using existing received PMU data. Once a missing data in current time step is detected,

DAPC can quickly complete the states using the predicted data and predict the future states for next time step. The specific application that we apply this mechanism for is wide-area oscillation damping, which following [17] is posed as a sparse linear quadratic regulator (LQR) design. Decentralized Kalman filters are assumed to be installed at every generator site to estimate the generator states through PMU measurements. Therefore, we use a recovery and prediction framework to guarantee states information exchanged between any pair of generators is complete even though some information is missing due to the wide-area communication delay or packet loss, thus the designed DAPC can continuously to ensure system stability and optimal performance. We illustrate the proposed method using the IEEE 39-bus power system model states data collected under power system toolbox (PST). Experimental results show that our framework can effectively guarantee the closed-loop performance for wide-area control.

5.2 Delay Aware Prediction Control Framework

Considering a power system with n -generator, and each generator is connected to a VM in the cloud. As stated in section 2.2, ideally, every VM needs to have full state x via data exchange with other generators, then they can compute their respective u_i , and send the signal back to generator i for actuation. In practice, if data loss or data delay happens, the VM cannot compute the control signal and send back to the generator which may significantly influence control performance or even cause the power system unstable. Thus, we proposed a Delay Aware Prediction Control framework (DAPC) to address this problem. The pseudo-code for DAPC is presented in Algorithm 1.

The framework equips each machine with a predictor shown in Figure 5.1 as an example. Each machine at time instance t_f , will store all states as initial states from other machine

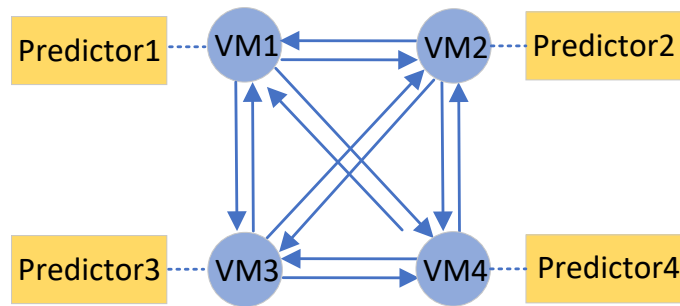


Figure 5.1 The typology of the DAPC

when each machine are fully observed. At time instance t_{f+1} , each machine predicts all states using its predictor and detect if the real states data have been received or not. If the states are NOT fully received from any machine, predicted sates are used to complete the missing states. The framework will perform similar steps for $t_{f+2}, t_{f+3}, \dots, t_{f+n}$. Figure 5.2 shows how DAPC completes the states at t_0 , assuming that the machine has full states at t_{-1} . The predictor will predict all states x_1, x_2, \dots, x_n for t_0 at t_{-1} . At t_0 , we detect that states from generator 2, generator 4 and generator $n - 1$ are not received on time due to network problems, then x_2, x_4 and x_{n-1} will be filled with the predicted values and other states will keep the measured data.

5.3 System Implementation

In the previous section, we discussed the general framework of DAPC. In this section, we will introduce two predictors for DAPC: a fuzzy time series forecasting model and a LSTM RNN model.

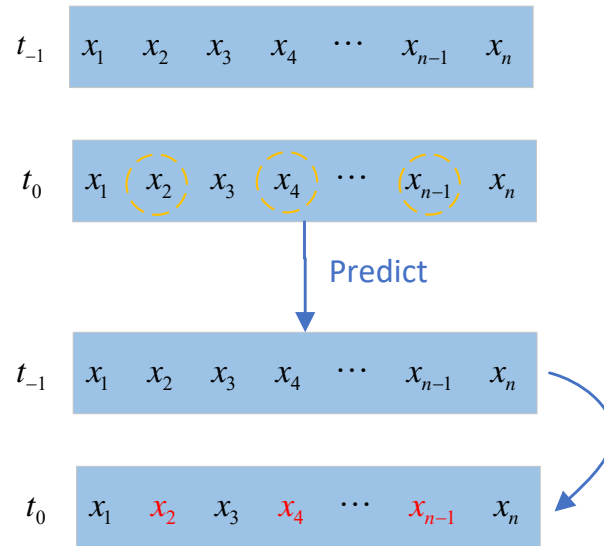


Figure 5.2 The example of DAPC states completion

Algorithm 3 DAPC

- 1: For each machine at time instance t_f , store all states as initial states from other machine when each machine are fully observed.
 - 2: Predict the states in each machine t_{f+1}
 - 3: **if** The states at t_{f+1} are fully received by each machine **then**
 - 4: Use actual states received to perform system control
 - 5: **else if** The states at t_{f+1} are NOT fully received by any machine **then**
 - 6: **for** each machine where the states are not complete received **do**
 - 7: Use predicted sates to complete the missing states
 - 8: Perform system control based on completed states
 - 9: Repeat Step 2-9 for $t_{f+2}, t_{f+3}, \dots, t_{f+n}$
-

5.3.1 Fuzzy Time Series

Song and Chissom [50]–[52] first introduced the concept of a fuzzy time series, based on the fuzzy set theory [63].

Definition 5.3.1. [63] Let $U = \{u_1, u_2, \dots, u_n\}$ be the universe of discourse. The fuzzy set A in the universe of discourse U can be defined as follows:

$$A = f_A(u_1)/u_1 + f_A(u_2)/u_2 + \dots + f_A(u_n)/u_n \quad (5.1)$$

where f_A is the membership function of the fuzzy set A , $f_A : U \rightarrow [0, 1]$, and $f_A(u_i)$ denotes the degree of membership of u_i belonging to the fuzzy set A , with $f_A(u_i) \in [0, 1]$ and $1 \leq i \leq n$.

Definition 5.3.2. [51] Let $Y(t) (t = \dots, 0, 1, 2, \dots)$, a subset of R (set of real numbers) be the universe of discourse in which fuzzy sets $f_i(t) (i = 1, 2, \dots)$ are defined. If $F(t)$ is a collection of $f_i(t) (i = 1, 2, \dots)$, then $F(t)$ is called a fuzzy time series on $Y(t) (t = \dots, 0, 1, 2, \dots)$.

Definition 5.3.3. [51] Let $F(t-1) = A_i$ and $F(t) = A_j$. The fuzzy logical relationship between $F(t-1)$ and $F(t)$ is referred by $A_i \rightarrow A_j$, where A_i is called the left-hand side and A_j is the right-hand side of the fuzzy logical relationship.

Definition 5.3.4. [13] Suppose there are fuzzy logical relationships with the same left-hand side A_i such that $A_i \rightarrow A_{j1}, A_i \rightarrow A_{j2}, \dots, A_i \rightarrow A_{jn}$. These fuzzy logical relationships can be grouped into a fuzzy logical relationship group, shown as $A_i \rightarrow A_{j1}, A_{j2}, \dots, A_{jn}$.

Definition 5.3.5. [42] Let $F(t-1) = A_i$ and $F(t) = A_j$, for $i, j = 1, 2, \dots, m$, which is denoted as $A_i \rightarrow A_j$. Then $k = j - i$ is the jump associated to this fuzzy logical relationship.

5.3.2 Fuzzy Time Series Forecasting Model

Over the past decade, many fuzzy time-series models have been proposed by following Song and Chissom's Definitions [50]–[52]. Among these models, Chen [13] proposed a model by applying simplified arithmetic operations in forecasting algorithms, replacing the complicated max-min composition operation introduced by Song and Chissom, and thus is more effective. Since we will modify Chen's model to a new weighted FTS, we first illustrate its procedure, as follows:

1. Partition the universe of discourse. First, according to the min and max values in the dataset, define the D_{min} and D_{max} variables and choose two arbitrary positive numbers, D_1 and D_2 , to partition the universe of discourse, i.e., $U = \{D_{min} - D_1, D_{max} + D_2\}$. Then divide this universe into equal length intervals, i.e., u_1, u_2, \dots, u_m .
2. Define the fuzzy sets and fuzzify the historical data. Using the defined sub intervals in former step, the fuzzy sets are defined as Eq. 5.2 and each historical datum in time series is fuzzified its corresponding fuzzy set accordingly.

$$A_i = \frac{0}{u_1} + \frac{0}{u_2} + \dots + \frac{0.5}{u_{i-1}} + \frac{1}{u_i} + \frac{0.5}{u_{i+1}} + \dots + \frac{0}{u_m}, i = 1, 2, 3, \dots m. \quad (5.2)$$

3. Establish fuzzy logic relationships (FLRs) and fuzzy logic relationship groups (FLRGs). FLRs are grouped based on the current states of the data according to Definition 5.3.4.
4. Calculate the forecast values. Let $F(t) = A_i$. If there is only one fuzzy logical relationship in the sequence, and $A_i \rightarrow A_j$, then $F(t + 1)$, the forecast output, equals A_j . If $A_i \rightarrow A_1, A_2, \dots, A_k$, then $F(t + 1)$ equals A_1, A_2, \dots, A_k . If there are no fuzzy relationship groups, then the forecast output equals A_i .

5. Defuzzify. If the forecast $F(t + 1) = A_{i_1}, A_{i_2}, \dots, A_{i_k}$, the forecast values at time $t+1$ are calculated as

$$Final(t + 1) = \frac{\sum_{j=1}^k M_{ij}}{k} \quad (5.3)$$

where $M_{i_1}, M_{i_2}, \dots, M_{i_j}$ are the defuzzified values of $A_{i_1}, A_{i_2}, \dots, A_{i_k}$, respectively, and $Final(t + 1)$ is the final forecast.

5.3.3 Long Short Terms Memory

LSTM is an artificial RNN architecture used in the field of deep learning. However, during back propagation, RNN suffers from the vanishing gradient problem that is the short-term memory problem. LSTM was created as a solution to short-term memory problem. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. A structure of LSTM is shown in Fig. 5.3 and Fig. 5.4. The major innovation of LSTM is its memory cell c_t which essentially acts as an accumulator of the state information. The cell is accessed, written and cleared by several self-parameterized controlling gates. Every time a new input comes, its information will be accumulated to the cell if the input gate i_t is activated. Also, the past cell status c_{t-1} could be "forgotten" in this process if the forget gate f_t is on. Whether the latest cell output c_t will be propagated to the final state h_t is further controlled by the output gate o_t . The advantage of using the memory cell and gates to control information flow is that the gradient will be trapped in the cell and be prevented from vanishing too quickly.

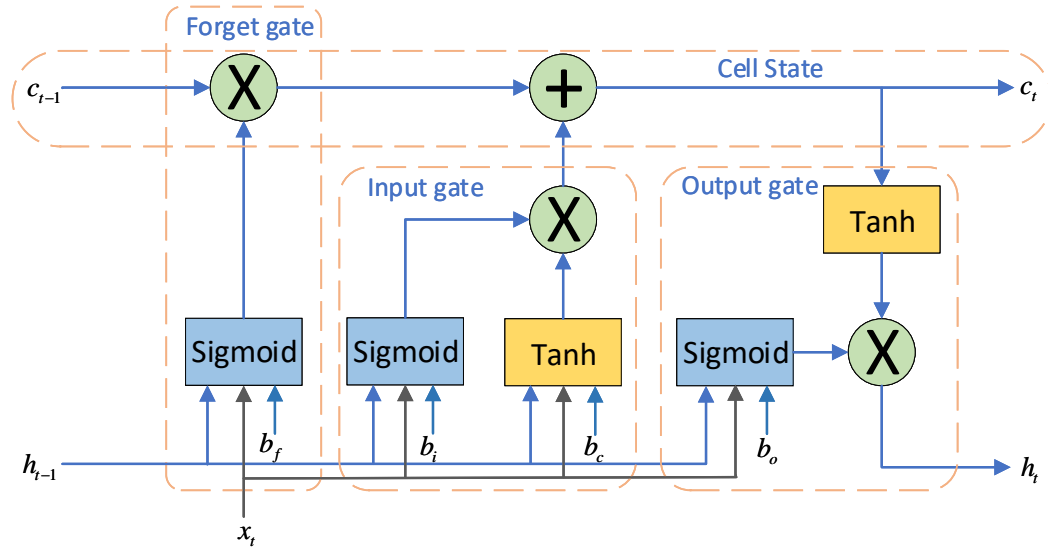


Figure 5.3 Structure of LSTM: A cell of LSTM.

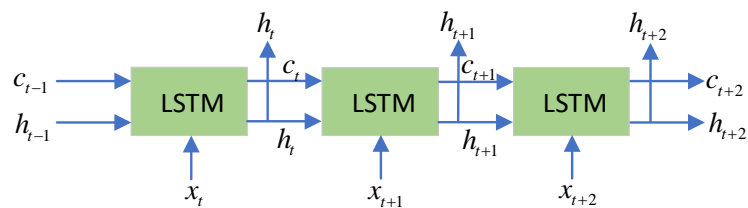


Figure 5.4 Structure of LSTM: Unrolled form of LSTM.

The LSTM memory cell is implemented as the following:

$$\begin{aligned}
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
 c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\
 h_t &= o_t \tanh(c_t)
 \end{aligned} \tag{5.4}$$

where σ is the logistic sigmoid function, and f , i , c , and o are the forget gate, input gate, cell vectors and output gate, all of which are the same size as the hidden vector h . The weight matrix subscripts have the meaning as the name suggests. For example, W_{xf} is input-forget gate matrix, W_{hi} is the hidden-input gate matrix etc.

5.3.4 The Workflow of LSTM Predictor

The workflow of predictor using the above LSTM is presented as follows:

1. Import the history data, i.e., a sequence of machine states for model training.
2. Normalize the data and frame it as supervised learning which in the form of $S(x_{t-1}) \rightarrow S(x_t)$.
3. Split the framed data into train and test sets with inputs (e.g., $S(x_{t-1})$) and outputs (e.g., $S(x_t)$).
4. Design the LSTM structure and determine the parameters such as the number of hidden nodes, densely-connected layer, batch size and the epoch number.
5. Train, validate and test predictor using the above processed data.

5.4 Simulation and Experiments

To validate the performance of the proposed DAPC and its practical implementation, we conduct the experiments on the IEEE 39-bus system. As the performance of LSTM (Keras) [34] is much impacted by the parameters, especially the number of hidden nodes (NHN), we first have a parameter study for LSTM. Specifically, we analyze the training loss and validation loss of different hidden nodes numbers as well as execution time of the proposed implementation. Second, we compare the prediction performance between the fuzzy predictor and the LSTM predictor. Finally, we apply both models to the system control.

All numerical simulations are conducted on a personal computer with an Intel Core i7-4790 CPU working at 3.4 GHz and 16 GB RAM. The proposed DAPC is implemented with Python 3.

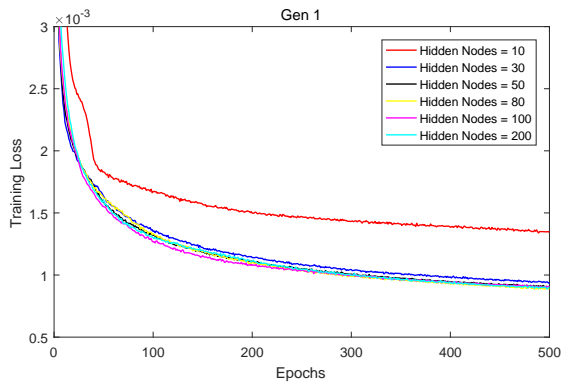
5.4.1 Data Generation

In the 39-bus 10-machine system, PMUs are installed on each machine then we have 10 observations. We assume that all PMUs measure 30 time-stamped system states in each second, and each one sends its measurements to the VM and exchange the measurements with each other with stochastic communication latency. We consider the system of a large number of random system faults which may potentially lead to the system stability issues. The system dynamics of each contingency is calculated using the PST software. PMU measurements are developed based on the calculated true system dynamics. Furthermore, we adopt the real latency values measured on the ExoGENI cloud testbed across the US. For PMU measurements with the same timestamp, their latencies are randomly selected from the latency values of all measured delays at an arbitrary time. All test cases are divided into

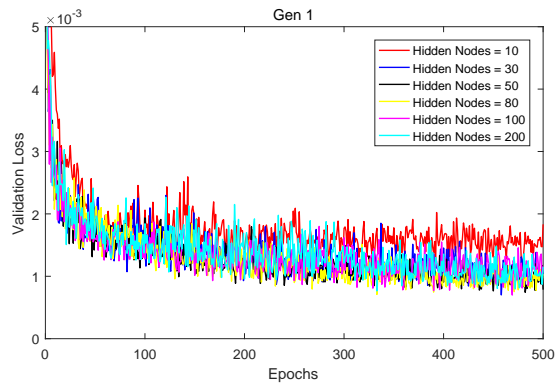
two groups with a 10:1 ratio. The former is used for training, while the latter for testing. While insufficient samples cannot characterize the system properties, too many samples can significantly increase the training time. In this work, the training and testing set sizes are 1000 and 100, respectively. The simulation results in the following sub-sections indicate that the selected sizes are appropriate.

5.4.2 Parameter Study of LSTM

To study the impact of NHN , we select it as 10, 30, 50, 80, 100 and 200 and draw both training loss and validation loss for Gen 1, 2, 5, 10. The batch size and epoch number are set to 100 and 500, respectively. Figure 5.5 — 5.8 show the training loss and validation loss convergence. As we can see from the figure, $NHN = 10$ is not enough since both training and validation are slowest to convergent. To investigate the difference between the different NHN setting. We further conduct a two-sample T-Test of Validation Loss for the four Generators. The results are shown in the Tables 5.1 — 5.4, where 1 represents statistical significance between two settings while 0 means there is no statistical significance. For example, for Gen 1, $NHN = 50$ has a statistical significance against $NHN = 10, 30, 100, 200$ but no statistical significance with $NHN = 80$. From the Tables 5.1 — 5.4, we observe that when $NHN \geq 80$, there is no performance improvement. In addition, considering the training time (Table 5.6), we select $NHN = 80$ as the final value for the predictor.

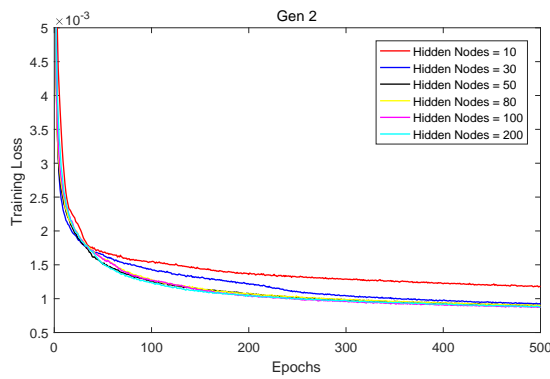


(a)

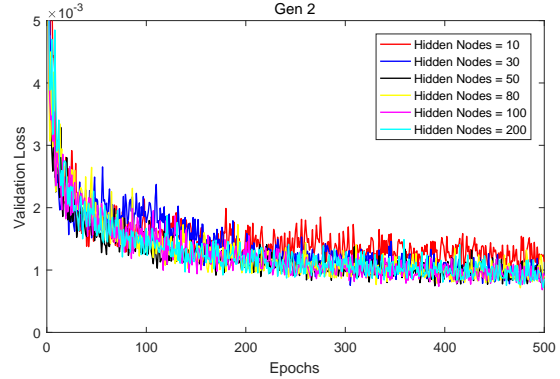


(b)

Figure 5.5 LSTM Loss of Gen 1: (a) Training loss. (b) Validation loss.



(a)



(b)

Figure 5.6 LSTM Loss of Gen 2: (a) Training loss. (b) Validation loss.

Table 5.1 Two-sample T-Test of Validation Loos for Gen 1 based on Different Hidden Nodes

Hidden Nodes Number	10	30	50	80	100	200
10	0	1	1	1	1	1
30	1	0	1	1	0	0
50	1	1	0	0	1	1
80	1	1	0	0	0	0
100	1	0	1	0	0	0
200	1	0	1	0	0	0

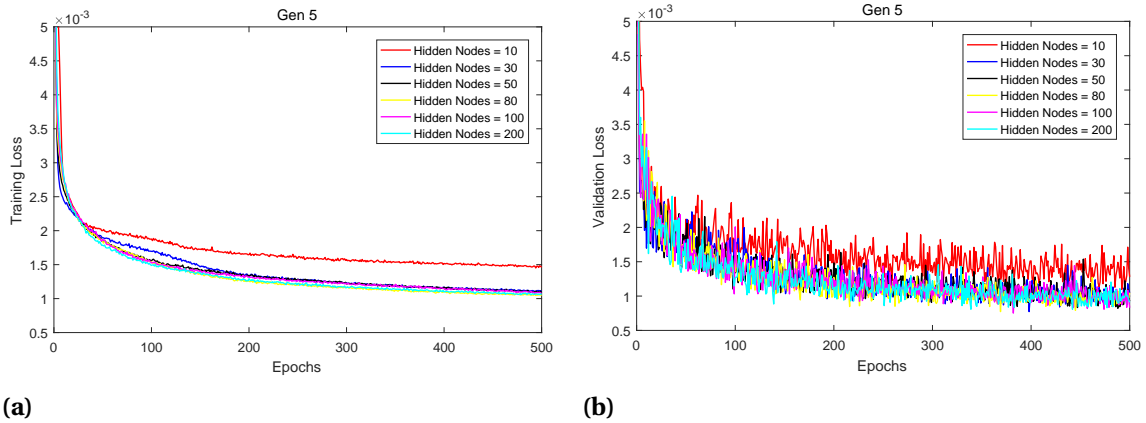


Figure 5.7 LSTM Loss of Gen 5: **(a)** Training loss. **(b)** Validation loss.

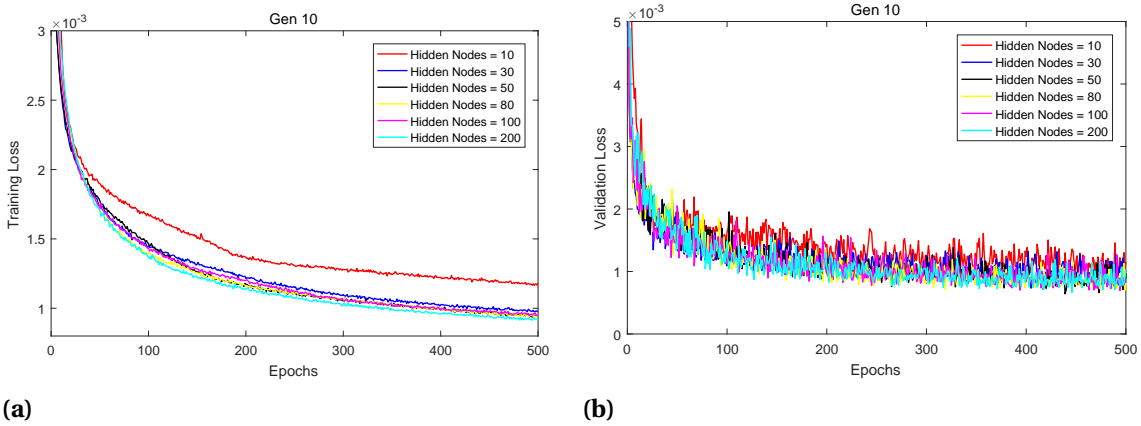


Figure 5.8 LSTM Loss of Gen 10: **(a)** Training loss. **(b)** Validation loss.

Table 5.2 Two-sample T-Test of Validation Loos for Gen 2 based on Different Hidden Nodes

Hidden Nodes Number	10	30	50	80	100	200
10	0	1	1	1	1	1
30	1	0	1	1	1	1
50	1	1	0	1	0	1
80	1	1	1	0	0	0
100	1	1	0	0	0	0
200	1	1	1	0	0	0

Table 5.3 Two-sample T-Test of Validation Loos for Gen 5 based on Different Hidden Nodes

Hidden Nodes Number	10	30	50	80	100	200
10	0	1	1	1	1	1
30	1	0	0	1	0	1
50	1	0	0	0	0	0
80	1	1	0	0	0	0
100	1	0	0	0	0	0
200	1	1	0	0	0	0

Table 5.4 Two-sample T-Test of Validation Loos for Gen 10 based on Different Hidden Nodes

Hidden Nodes Number	10	30	50	80	100	200
10	0	1	1	1	1	1
30	1	0	0	1	0	0
50	1	0	0	0	0	0
80	1	1	0	0	0	0
100	1	0	0	0	0	0
200	1	0	0	0	0	0

5.4.3 Prediction accuracy and training time comparison

In this section, to evaluate the prediction performance, we apply both fuzzy and LSTM predictors to predict the missing states. Generally, time series forecasting describes predicting the observation at the next time step. This is called one-step forecast, as only one time step is to be predicted. We use the recursive strategy for the multi-step forecast for the prediction, which use a one-step model multiple times where the prediction for the prior time step is used as an input for making a prediction on the following time step. Figure 5.9

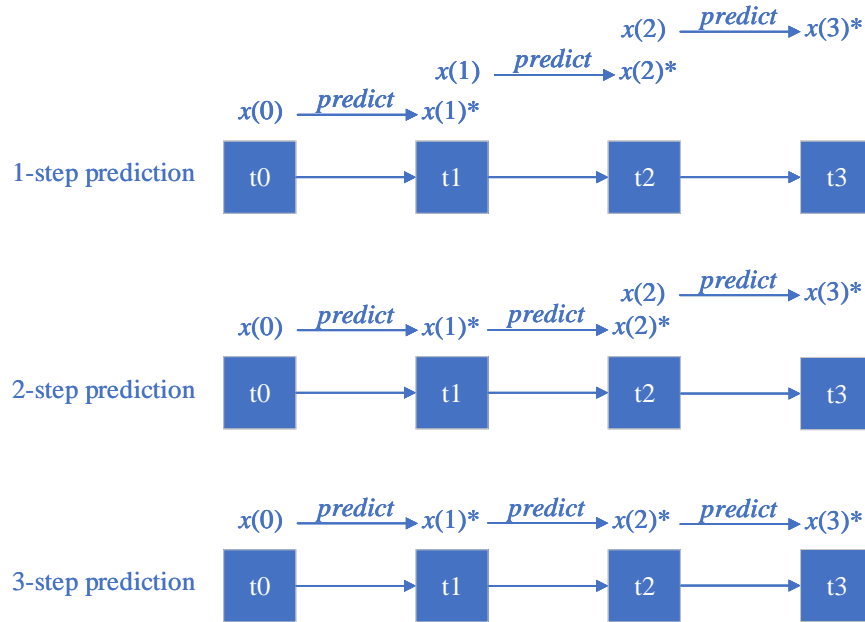


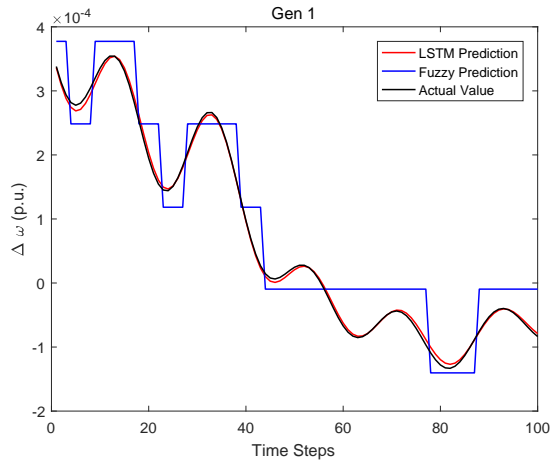
Figure 5.9 Prediction Steps Clarification.

illustrates multi-step prediction for this chapter. Given the observed states $x(0)$ at time t_0 , a 1-step forecast predicts the states $x(1)^*$ at time t_1 , and use the true states $x(1)$ to predict states $x(2)^*$ at time t_2 . One-step prediction means the states can arrive within 2-step delay in practical network transmission. For the 2-step prediction, we predict states $x(1)^*$ at t_1 using the given states $x(0)$ at t_0 , and predict states $x(2)^*$ at t_2 by the predicted value $x(1)^*$. Then $x(3)^*$ can be predicted by the true states $x(2)$. That means the states can have a 2-step data missing in practical network transmission. The 3-step prediction, 4-step prediction, ..., can be done in the same manner. Figure 5.10 shows the one-step prediction results and the actual value and Table 5.5 lists the RMSE for the two models.

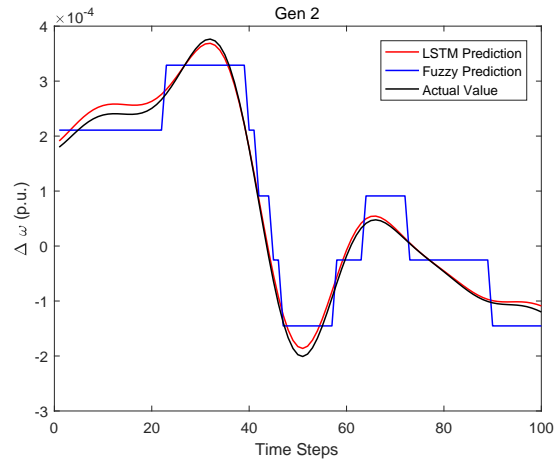
Table 5.5 The comparison of RMSE for different models against actual values

Gen	LSTM Prediction	Fuzzy Prediction
1	2.2657e-4	2.5124e-4
2	7.5617e-5	1.6559e-4
5	1.3632e-4	1.8587e-4
10	2.3742e-4	2.5530e-4

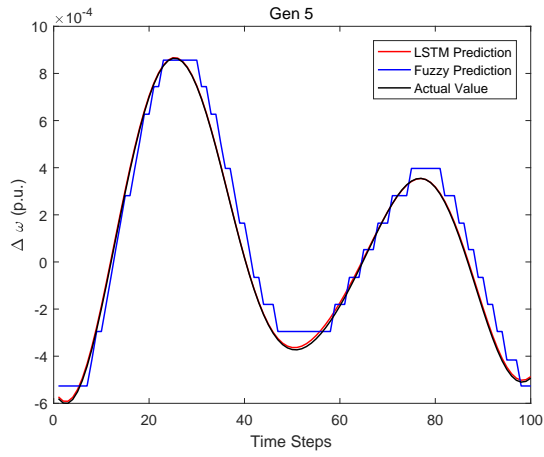
As can be seen from the Figure 5.10 and Table 5.5, the LSTM predictor has an excellent prediction performance when compared with fuzzy predictor. However, when consider the training CPU time which are listed in Table 5.6, the fuzzy predictor has its own advantage in the dynamic situation that needs to quickly adapt the model.



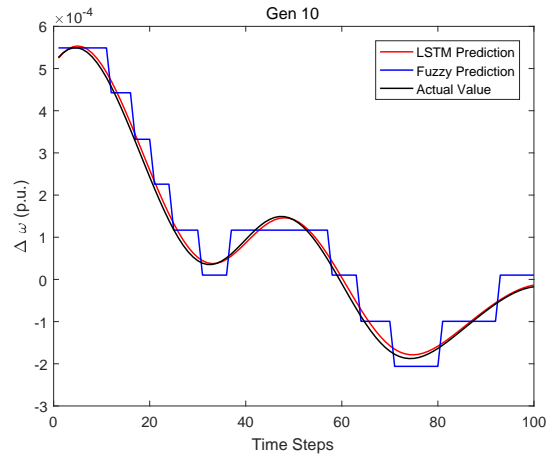
(a)



(b)



(c)



(d)

Figure 5.10 Prediction accuracy comparison

Table 5.6 Average Time of Training and Prediction for Different LSTM Hidden Nodes and Fuzzy Model

Hidden Nodes Number	Average Time for Training of One Epoch(s)	Average Time for Predicting One Future State (ms)
10	6	0.42
30	7	0.45
50	9	0.50
80	12	0.51
100	18	0.52
200	32	0.63
Fuzzy Model	0.001	0.001

5.4.4 Control performance validation and comparison

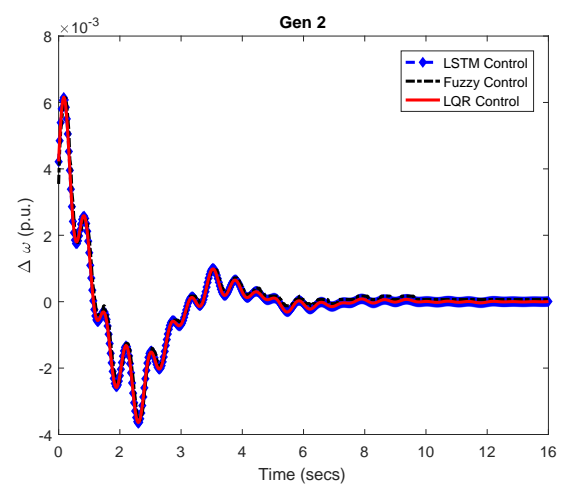
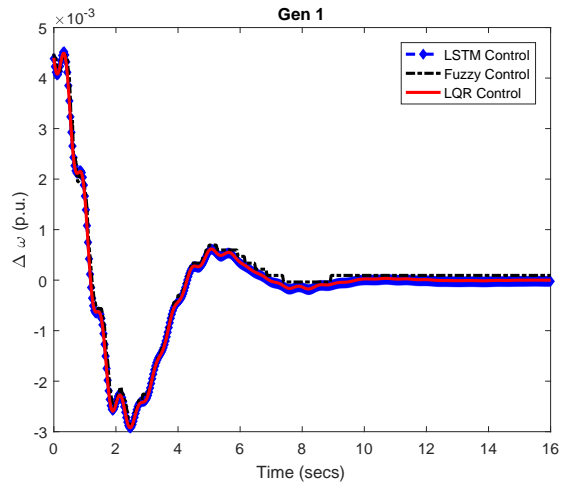
In this section, to validate and compare the control performance of DAPC under different network conditions, we simulate the experiments on 6 different cases shown in Table 5.7. These cases represent various network status from "best" (Case 1) to "worst" (Case 6). Specifically, in case 1, all PMU data will arrive no later than 2 time steps so that DAPC can just one step prediction to guarantee the best performance. Case 6 is the worst case, where only 10% PMU data arrives in 2 time steps; 30% data arrives in 3 time steps; 20% data arrives in 4 time steps; 20% data arrives in 5 time steps; 10% data arrives in 6 time steps and 10% data arrives in 7 steps. As can be seen from the table, we list the n -step prediction used in

different scenarios.

Table 5.7 6 Cases for validating the control performance

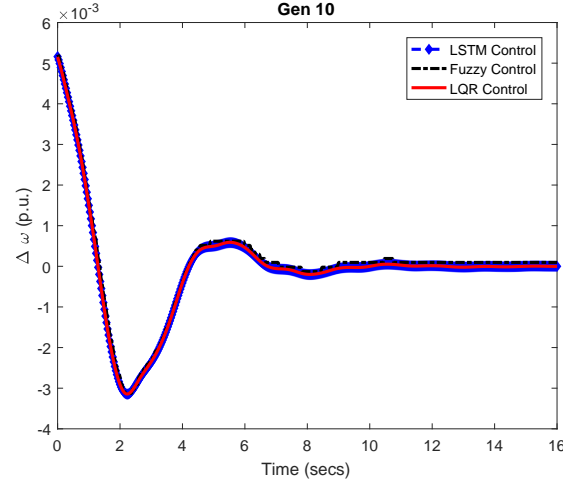
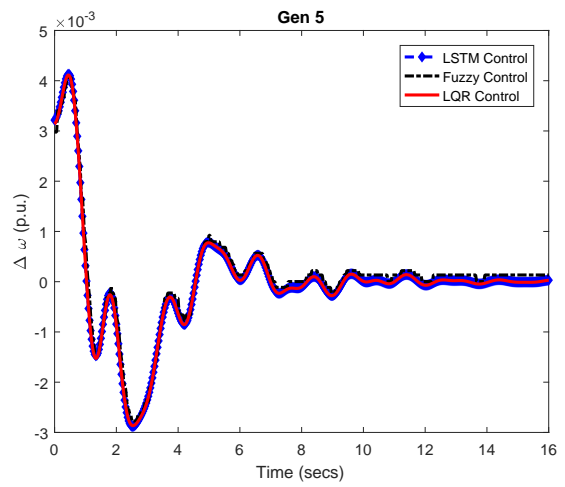
Cases	1-step	2-step	3-step	4-step	5-step	6-step
Case 1	100%	0%	0%	0%	0%	0%
Case 2	90%	4%	2%	2%	1%	1%
Case 3	80%	10%	4%	3%	2%	1%
Case 4	60%	20%	10%	5%	3%	2%
Case 5	40%	25%	15%	10%	5%	5%
Case 6	10%	30%	20%	20%	10%	10%

Figures 5.11-5.16 show prediction results for the closed-loop responses of generator1, 2, 5 and 10. We compare LSTM based prediction control with the Fuzzy based prediction control and also with LQR control which is an ideal case without network delay. The results for Case 1 in Fig. 5.11 show that LSTM based prediction control performs better than Fuzzy based case as it almost overlapping the curves of LQR control. Then further experiments are tested by adding more multi-step predictions, which means the system encounters more and more delays. Figs. 5.12-5.16 indicate that fuzzy based prediction control become worse and worse as more delay are added to the data transmission. However, LSTM based prediction control can still keep close to the ideal LQR control performance even in Case 6 that 90% of the data arrived with delay. We didn't include fuzzy predictor results for Case 5 and Case 6 as the performance of fuzzy prediction are too bad. The results reveal that the proposed DAPC framework with LSTM predictor is promising in the real applications.



(a)

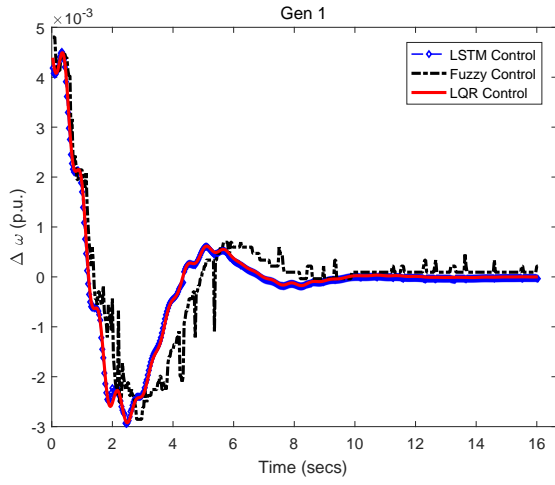
(b)



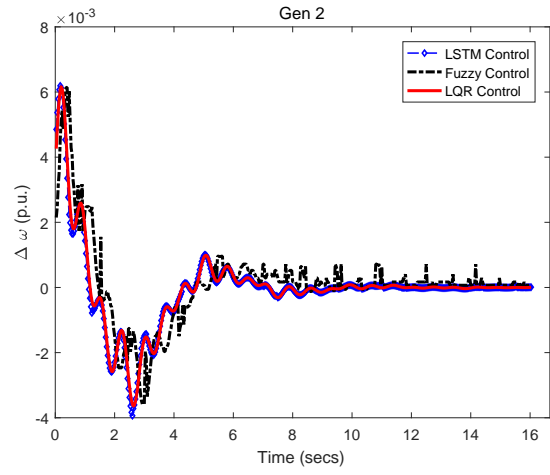
(c)

(d)

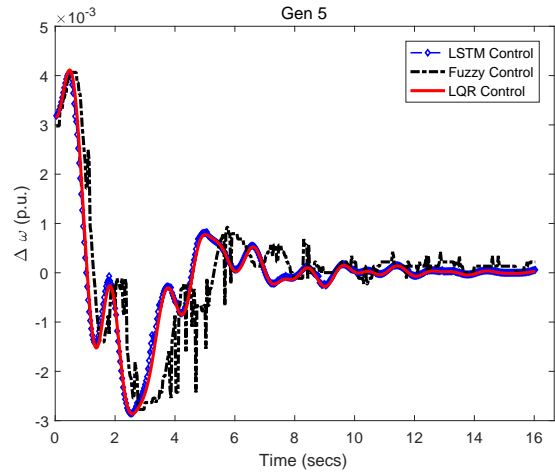
Figure 5.11 Performance of Case 1 in damping oscillations



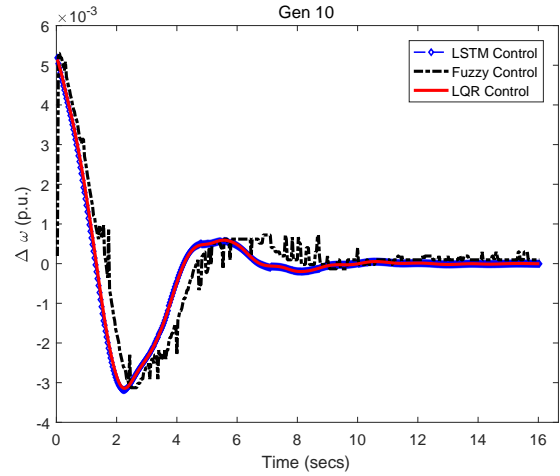
(a)



(b)

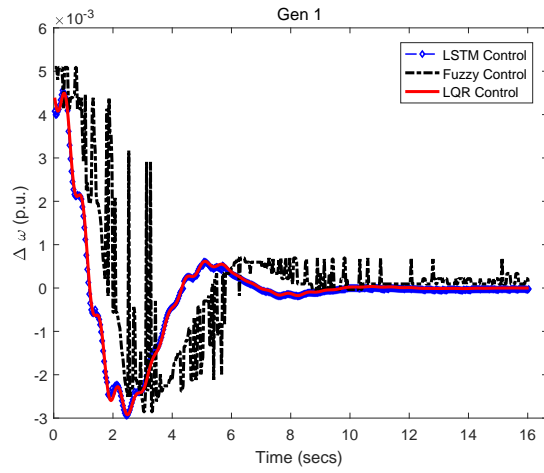


(c)

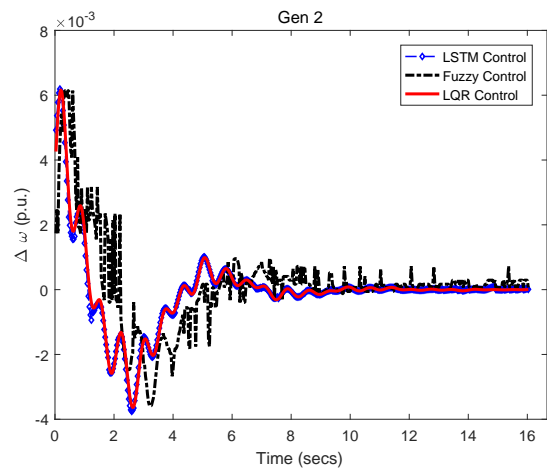


(d)

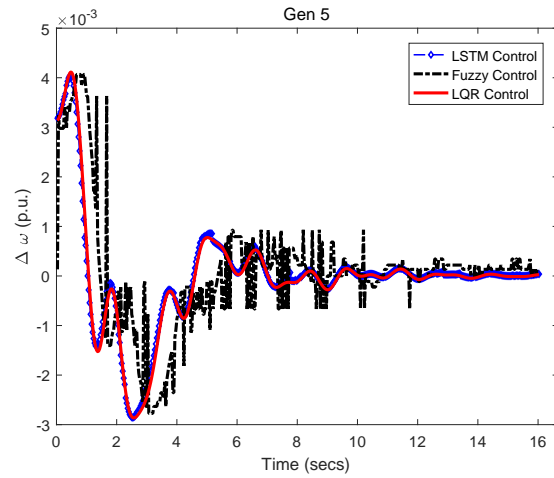
Figure 5.12 Performance of Case 2 in damping oscillations



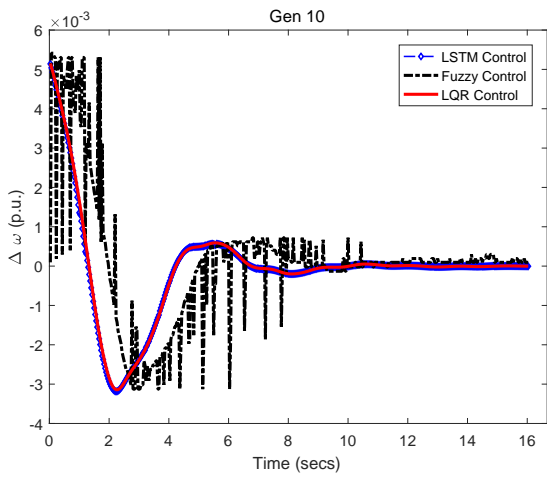
(a)



(b)

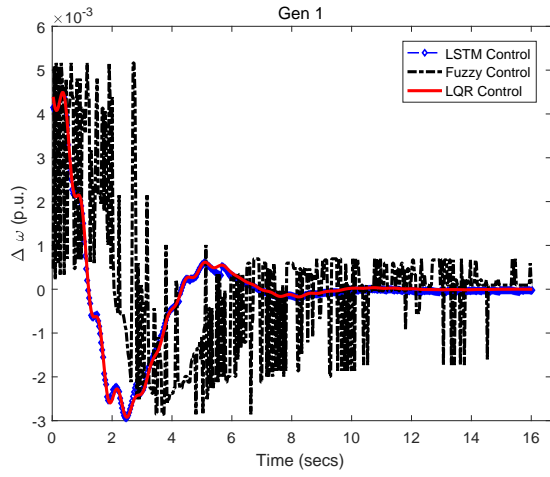


(c)

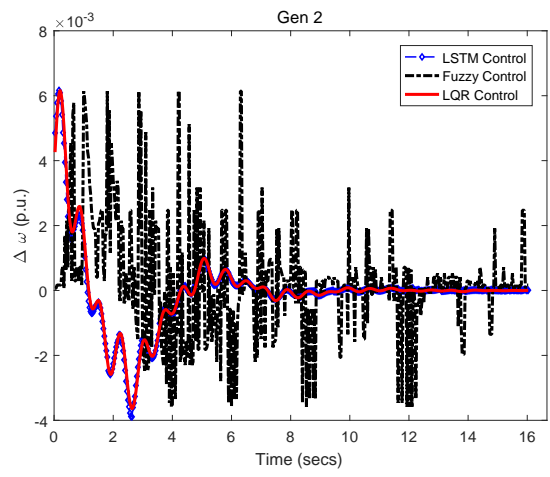


(d)

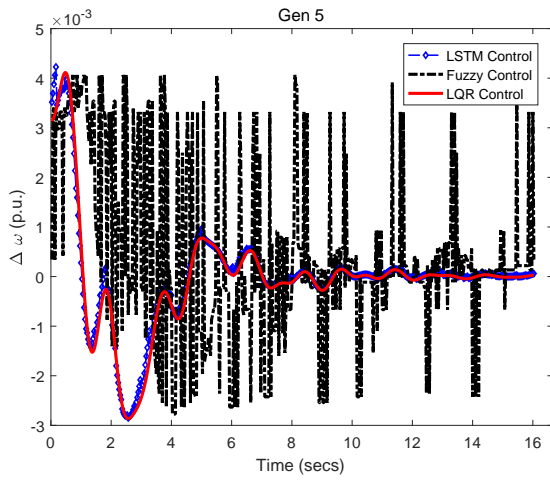
Figure 5.13 Performance of Case 3 in damping oscillations



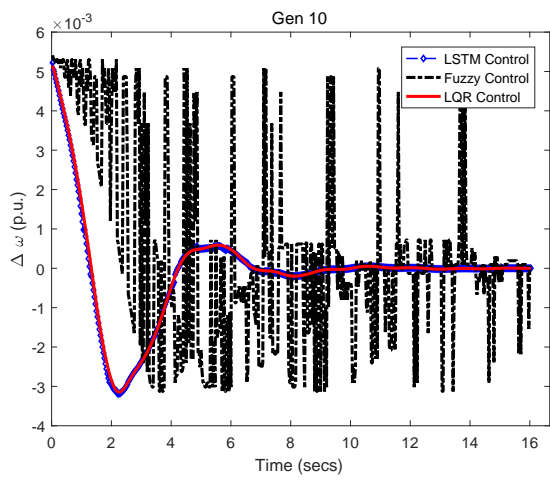
(a)



(b)

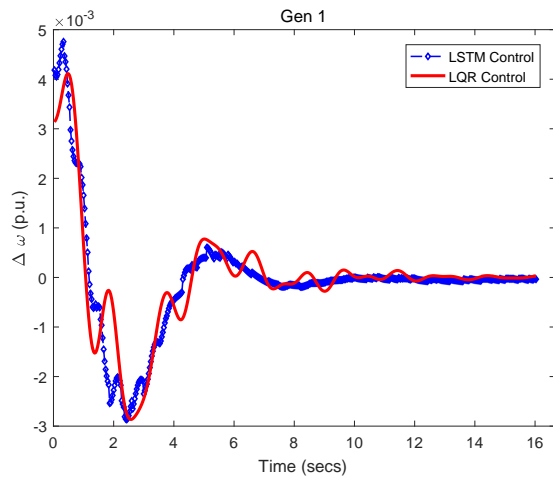


(c)

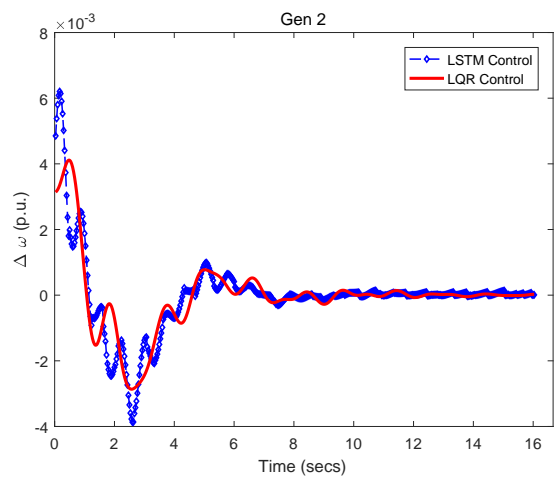


(d)

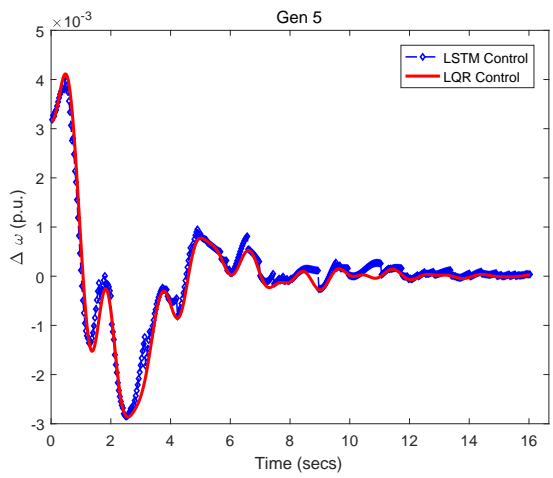
Figure 5.14 Performance of Case 4 in damping oscillations



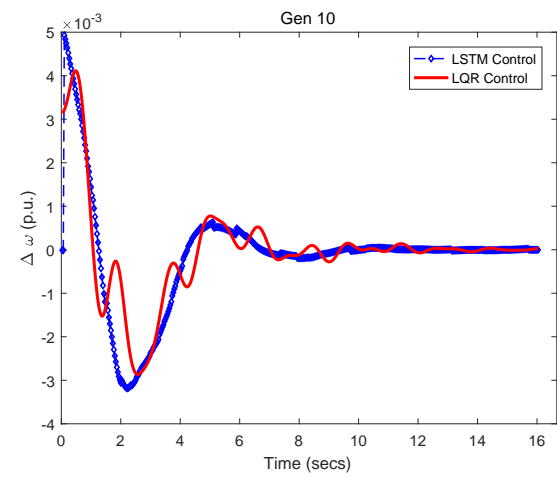
(a)



(b)

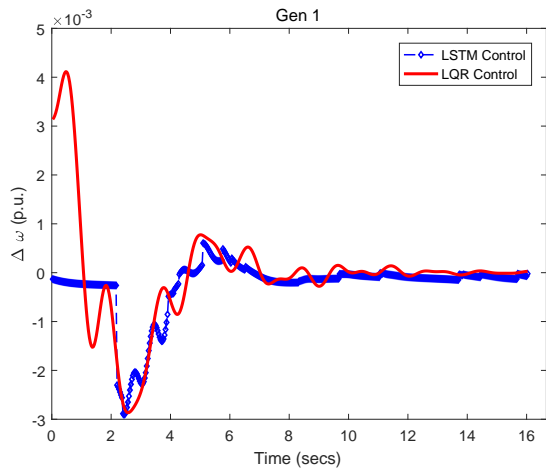


(c)

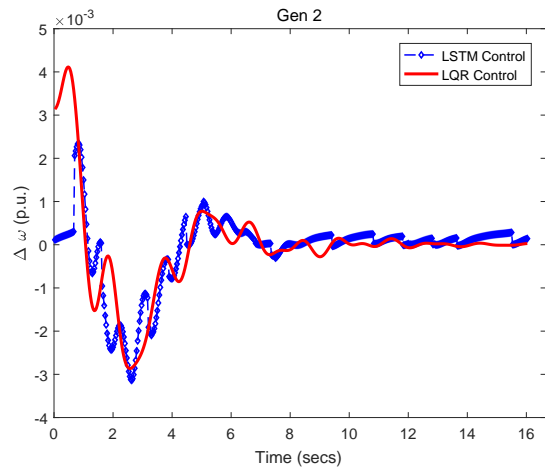


(d)

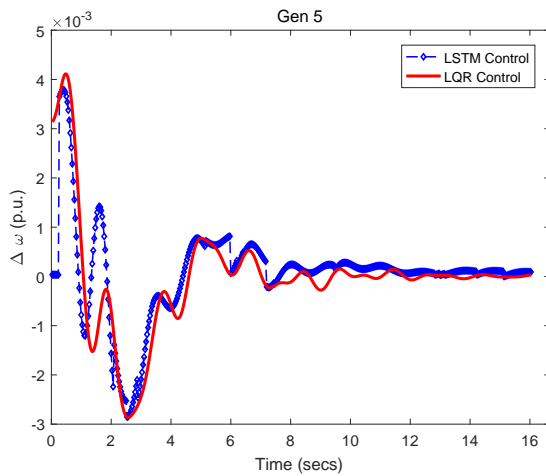
Figure 5.15 Performance of Case 5 in damping oscillations



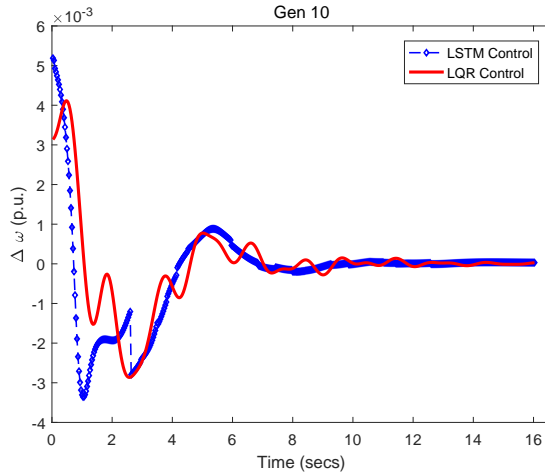
(a)



(b)



(c)



(d)

Figure 5.16 Performance of Case 6 in damping oscillations

5.5 Conclusions

In this paper we propose a novel delay aware prediction control framework to address the data transmission delay of PMU measurements in modern power systems. As typical emergent power system data-centric applications can suffer from communication latency, the proposed framework aims to recover real-time power system states and predict future states. The framework uses predictor to manipulate available synchrophasor data and recover the unknown ones. To demonstrate the efficacy of the proposed framework, we implemented DAPC employing recent advanced machine learning techniques. A series of simulations are conducted on the IEEE 39-bus power system, and the results demonstrate that DAPC can promptly recover the incomplete states and predict the future states online and thereby guarantee power system stability of the proposed implementation.

CHAPTER

6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

With the increasing complexity in the interconnection of power networks, stability of the system has been studied extensively. One of the biggest challenges for wide-area control of power systems is the need for a highly robust communication system that works in sync with the control functionalities. The majority of the ongoing NASPInet research are devoted

to the hardware and architectural planning aspects of wide-area communication, with little attention to understanding how complicated multiple-input-multiple-output (MIMO) control loops, when implemented on top of the existing communication protocols, may perform under various operating uncertainties. For example, the exponentially increasing number of PMUs, producing Terabytes of real-time data that need to be communicated from remote locations to the control centers, may cause a high possibility of network congestion. The network delays in the wide-area communication, however, may cause the closed-loop performance of the grid to degrade. To address the above challenges, in our first study, we propose the adoption of a simple latency control algorithm implemented in a software-defined network (SDN) connecting the VMs in the wide-area cloud by which delays in the control loop can always be optimized in order to retain their stability limit. The algorithm uses a greedy routing path selection based on active latency measurement for data transfer, and can be run periodically to pick the best available routes while the wide-area control loop is running.

Considering perfect delay bounds can almost never be guaranteed no matter how robust the SDN controllers are, in which case the best option might be to drop that link from the control loop while still ensuring that the closed-loop performance remains close to optimal. In our second study, we present a sparsity-constrained LQR algorithm that starts from an ideal all-to-all connected communication network, and progressively drops links if the traffic in those links are predicted to be high. Each drop is accompanied with a simultaneous optimal tuning of the control gains associated with the existing links using a method called Geromel's algorithm.

However, the more links dropped, the worse the control performance is. In practical situation, the network delay is not constant but time-varying, therefore, it is obviously not optimal if keep dropping links but never bringing back the dropped links even if the network

recovers, where the control performance is limited. To address this issue, in our third study, we propose a kernel density estimation (KDE) based model to dynamically detect adverse network delay area by monitoring the round-trip delay of interconnected links. Once a link is detected entering or out the delay area, we simultaneously tune the control gains associated with the existing links so that the overall closed-loop performance of the wide-area control system remains approximated optimal. We illustrate the proposed method using the real network data collected from ExoGENI cloud platform and simulations of the IEEE 39-bus power system model. Experimental results show that our framework can effectively guarantee the closed-loop performance for wide-area control in the time-varying network traffics.

In practice, tuning the control gains following delay detection has a very high computing requirement for the control center. If the control center cannot compute the feedback gains within the stability limit time, the system could be unstable as well. Hence, in our fourth study, we present a novel delay aware prediction control framework to address the problem of missing power system state variables due to the existence of communication latency in wide-area measurement control systems. This capability is particularly essential for dynamic power system scenarios where fast remedial control actions are required due to system events or faults. In this work, a PMU data prediction framework and its practical implementation in power system damping control are proposed to predict future states using the existing PMU data and to complete missing variables with the predicted data. The framework establishes an online prediction scheme, and the proposed implementation adopts recent machine learning advances in data processing. Simulation results indicate that the proposed framework has superior accuracy and is computational efficient, and fulfill the control requirements for power system under delayed network condition.

6.2 Future Work

In our future work, considering the increasing complexity of power systems, it becomes more challenging to get the accurate system model, which causes that the traditional model-based control theories and methods are impractical for control issues. Thus, we propose to further investigate big data and machine learning techniques and tools to help address some critical issues like usability, accuracy and confidence. We will also develop the applications for power system analytics in fault detection and on-line, real-time dynamic security assessment (DSA). In addition, we will present the applications for power system operation on outage management and cyber security.

BIBLIOGRAPHY

- [1] N. T. Anh, L. Vanfretti, J. Driesen & D. Van Hertem, “A quantitative method to determine ict delay requirements for wide-area power system damping controllers”, *IEEE Transactions on Power Systems*, **30**, no. 4, pp. 2023–2030, 2014.
- [2] R. Arastoo, N. Motee & M. V. Kothare, “Optimal sparse output feedback control design: A rank constrained optimization approach”, *arXiv preprint arXiv:1412.8236*, 2014.
- [3] D. Bian, M. Kuzlu, M. Pipattanasomporn & S. Rahman, “Analysis of communication schemes for advanced metering infrastructure (ami)”, *2014 IEEE PES General Meeting/ Conference & Exposition*, IEEE, 2014, pp. 1–5.
- [4] G. E. Boukarim, S. Wang, J. H. Chow, G. N. Taranto & N. Martins, “A comparison of classical, robust, and decentralized control designs for multiple power system stabilizers”, *IEEE Transactions on Power Systems*, **15**, no. 4, pp. 1287–1292, 2000.
- [5] D. Cai, P. Regulski, M. Osborne & V. Terzija, “Wide area inter-area oscillation monitoring using fast nonlinear estimation algorithm”, *IEEE Transactions on Smart Grid*, **4**, no. 3, pp. 1721–1731, 2013.
- [6] A. Chakraborty & P. Khargonekar, “Introduction to wide-area monitoring and control”, *American Control Conference*, 2013, pp. 6758–6770.
- [7] A. Chakraborty, “Wide-area damping control of power systems using dynamic clustering and tcsc-based redesigns”, *IEEE_{JSG}*, **3**, no. 3, pp. 1503–1514, 2012.
- [8] A. Chakraborty & Y. Xin, “Hardware-in-the-loop simulations and verifications of smart power systems over an exo-geni testbed”, *2013 Second GENI Research and Educational Experiment Workshop*, GREE-2013, 2013.
- [9] B. Chaudhuri, R. Majumder & B. C. Pal, “Wide-area measurement-based stabilizing control of power system considering signal transmission delay”, *IEEE Transactions on Power Systems*, **19**, no. 4, pp. 1971–1979, 2004.
- [10] N. R. Chaudhuri, D. Chakraborty & B. Chaudhuri, “Damping control in power systems under constrained communication bandwidth: A predictor corrector strategy”, *IEEE_{JPRS}*, **20**, no. 1, pp. 223–231, 2012.
- [11] N. R. Chaudhuri, B. Chaudhuri, S. Ray & R. Majumder, “Wide-area phasor power oscillation damping controller: A new approach to handling time-varying signal latency”, *IET generation, transmission & distribution*, **4**, no. 5, pp. 620–630, 2010.

- [12] G. Chavan, M. Weiss, A. Chakraborty, S. Bhattacharya, A. Salazar & F.-H. Ashrafi, “Identification and predictive analysis of a multi-area wecc power system model using synchrophasors”, *IEEE Transactions on Smart Grid*, **8**, no. 4, pp. 1977–1986, 2016.
- [13] S. Chen, “Forecasting enrollments based on fuzzy time series”, *Fuzzy sets and systems*, **81**, no. 3, pp. 311–319, 1996.
- [14] Y. Chen, L. Xie & P. Kumar, “Dimensionality reduction and early event detection using online synchrophasor data”, *2013 IEEE Power & Energy Society General Meeting*, IEEE, 2013, pp. 1–5.
- [15] T. Chin, M. Rahouti & K. Xiong, “End-to-end delay minimization approaches using software-defined networking”, pp. 184–189, 2017.
- [16] F. De Turck, P. Chemouil, R. Boutaba, M. Yu, C. E. Rothenberg & K. Shiimoto, “Guest editors’ introduction: Special issue on management of softwarized networks”, *IEEE Transactions on Network and Service Management*, **13**, no. 3, pp. 362–365, 2016.
- [17] F. Dörfler, M. R. Jovanović, M. Chertkov & F. Bullo, “Sparsity-promoting Optimal Wide-area Control of Power Networks”, *IEEE Transactions on Power Systems*, **29**, no. 5, pp. 2281–2291, 2014.
- [18] *ExoGENI: www.exogeni.net*.
- [19] A. Fernandez, R. B. H. Bunke & J. Schmidhuber, “A novel connectionist system for improved unconstrained handwriting recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**, no. 5, 2009.
- [20] P. Gao, M. Wang, S. G. Ghiocel, J. H. Chow, B. Fardanesh & G. Stefopoulos, “Missing data recovery by exploiting low-dimensionality in power system synchrophasor measurements”, *IEEE Transactions on Power Systems*, **31**, no. 2, pp. 1006–1013, 2015.
- [21] A. Gijare, “Software-defined networking: Technologies and global markets”, 2016.
- [22] S. Hochreiter & J. Schmidhuber, “Long short-term memory”, *Neural computation*, **9**, no. 8, pp. 1735–1780, 1997.
- [23] A. J. Izenman, “Review papers: Recent developments in nonparametric density estimation”, *Journal of the American Statistical Association*, **86**, no. 413, pp. 205–224, 1991.

- [24] A. Jain, A. Chakraborty & E. Biyik, “An Online Structurally Constrained LQR Design for Damping Oscillations in Power System Networks”, *American Control Conference (ACC)*, IEEE, 2017, pp. 2093–2098.
- [25] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll & P. Tran-Gia, “Modeling and performance evaluation of an openflow architecture”, *Proceedings of the 23rd international teletraffic congress*, International Teletraffic Congress, 2011, pp. 1–7.
- [26] M. Karakus & A. Durresi, “A Survey: Control Plane Scalability Issues and Approaches in Software-defined Networking (SDN)”, *Computer Networks*, **112**, pp. 279–293, 2017.
- [27] D. Kreutz, F. Ramos & P. Verissimo, “Towards Secure and Dependable Software-defined Networks”, *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM, 2013, pp. 55–60.
- [28] P. Kundur, *Power system stability and control*. McGraw-hill New York, 1994.
- [29] A. E. Leon & J. A. Solsona, “Power oscillation damping improvement by adding multiple wind farms to wide-area coordinating controls”, *IEEE Transactions on Power Systems*, **29**, no. 3, pp. 1356–1364, 2013.
- [30] X. Li & X. Wu, “Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition”, *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 4520–4524.
- [31] F. Lian, A. Chakraborty & A. Duel-Hallen, “Game-theoretic multi-agent control and network cost allocation under communication constraints”, *IEEE journal on selected areas in communications*, **35**, no. 2, pp. 330–340, 2017.
- [32] F. Lin, M. Fardad & M. R. Jovanović, “Design of optimal sparse feedback gains via the alternating direction method of multipliers”, *IEEE Transactions on Automatic Control*, **58**, no. 9, pp. 2426–2431, 2013.
- [33] Q. Liu, V. Vittal & N. Elia, “Lpv supplementary damping controller design for a thyristor controlled series capacitor (tcsc) device”, *IEEE transactions on power systems*, **21**, no. 3, pp. 1242–1249, 2006.
- [34] *LSTM (Keras)*: www.tensorflow.org/guide/keras/rnn.
- [35] C. Lu, X. Zhang, X. Wang & Y. Han, “Mathematical expectation modeling of wide-area controlled power systems with stochastic time delay”, *IEEE transactions on Smart Grid*, **6**, no. 3, pp. 1511–1519, 2015.

- [36] H. Ni, A. Chakraborty & Y. Xin, “Online Tuning of Cloud-based Wide-Area Controllers with Variations in Network Traffic”, *Power & Energy Society General Meeting, 2019 IEEE*.
- [37] H. Ni, M. Rahouti, A. Chakraborty, K. Xiong & Y. Xin, “A Distributed Cloud-based Wide-Area Controller with SDN-Enabled Delay Optimization: <https://goo.gl/BDzJft>”, *Power & Energy Society General Meeting, 2018 IEEE*.
- [38] *North American Synchronphasor Initiative: www.naspi.org*.
- [39] B. P. Padhy, S. C. Srivastava & N. K. Verma, “A wide-area damping controller considering network input and output delays and packet drop”, *IEEE Transactions on Power Systems*, **32**, no. 1, pp. 166–176, 2016.
- [40] S. Pal, B. Sikdar & J. Chow, “Real-time detection of packet drop attacks on synchronphasor data”, *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, 2014, pp. 896–901.
- [41] A. G. Phadke & J. S. Thorp, *Synchronized phasor measurements and their applications*. Springer, 2008, vol. 1.
- [42] A. Rubio, J. D. Bermúdez & E. Vercher, “Improving stock index forecasts by using a new weighted fuzzy-trend time series method”, *Expert Systems with Applications*, **76**, pp. 12–20, 2017.
- [43] T. Sadamoto, A. Chakraborty, T. Ishizaki & J.-i. Imura, “Dynamic modeling, stability, and control of power systems with distributed energy resources: Handling faults using two control methods in tandem”, *IEEE Control Systems Magazine*, **39**, no. 2, pp. 34–65, 2019.
- [44] H. Sak, A. W. Senior & F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling”, 2014.
- [45] Y. Seyedi, H. Karimi & J. M. Guerrero, “Centralized disturbance detection in smart microgrids with noisy and intermittent synchronphasor data”, *IEEE Transactions on Smart Grid*, **8**, no. 6, pp. 2775–2783, 2016.
- [46] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller & N. Rao, “Are We Ready for SDN? Implementation Challenges for Software-Defined Networks”, *IEEE Communications Magazine*, **51**, no. 7, pp. 36–43, 2013.

- [47] S. Sheng, K. Li, W. Chan, X. Zeng, D. Shi & X. Duan, “Adaptive agent-based wide-area current differential protection system”, *IEEE Transactions on Industry Applications*, **46**, no. 5, pp. 2111–2117, 2010.
- [48] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.
- [49] A. K. Singh & B. C. Pal, “Decentralized dynamic state estimation in power systems using unscented transformation”, *IEEE Transactions on Power Systems*, **29**, no. 2, pp. 794–804, 2014.
- [50] Q. Song & B. S. Chissom, “Forecasting enrollments with fuzzy time series—part i”, *Fuzzy sets and systems*, **54**, no. 1, pp. 1–9, 1993.
- [51] Q. Song & B. S. Chissom, “Fuzzy time series and its models”, *Fuzzy sets and systems*, **54**, no. 3, pp. 269–277, 1993.
- [52] Q. Song & B. S. Chissom, “Forecasting enrollments with fuzzy time series—part ii”, *Fuzzy sets and systems*, **62**, no. 1, pp. 1–8, 1994.
- [53] D. Soudbakhsh, A. Chakraborty & A. Annaswamy, “A delay-aware cyber-physical architecture for wide-area control of power systems”, *Control Engineering Practice*, **60**, pp. 171–182, 2017.
- [54] D. Soudbakhsh, A. Chakraborty & A. M. Annaswamy, “Delay-aware co-designs for wide-area control of power grids”, *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 2493–2498.
- [55] J. W. Stahlhut, T. J. Browne, G. T. Heydt & V. Vittal, “Latency viewed as a stochastic process and its impact on wide area power system control signals”, *IEEE Transactions on Power Systems*, **23**, no. 1, pp. 84–91, 2008.
- [56] K. Tomsovic, D. E. Bakken, V. Venkatasubramanian & A. Bose, “Designing the next generation of real-time control, communication, and computations for large power systems”, *Proceedings of the IEEE*, **93**, no. 5, pp. 965–979, 2005.
- [57] B. A. Turlach, “Bandwidth selection in kernel density estimation: A review”, *CORE and Institut de Statistique*, Citeseer, 1993.
- [58] S. Wang, W. Gao & A. S. Meliopoulos, “An alternative method for power system dynamic state estimation based on unscented transform”, *IEEE transactions on power systems*, **27**, no. 2, pp. 942–950, 2011.

- [59] Y. Wu, L. Nordström & D. E. Bakken, “Effects of bursty event traffic on synchrophasor delays in ieeec37.118, iec61850, and iec60870”, *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, 2015, pp. 478–484.
- [60] M. Wytock & J. Z. Kolter, “A fast algorithm for sparse controller design”, *arXiv preprint arXiv:1312.4892*, 2013.
- [61] W. Yao, L. Jiang, J. Wen, Q. Wu & S. Cheng, “Wide-area damping controller of facts devices for inter-area oscillations considering communication time delays”, *IEEE Transactions on Power Systems*, **29**, no. 1, pp. 318–329, 2013.
- [62] T. Zabaiou, L.-A. Dessaint, F.-A. Okou & R. Grondin, “Wide-area coordinating control of svcs and synchronous generators with signal transmission delay compensation”, *IEEE PES General Meeting*, IEEE, 2010, pp. 1–9.
- [63] L. A. Zadeh, “Fuzzy sets”, *Information and control*, **8**, no. 3, pp. 338–353, 1965.
- [64] J. Zhang, S. Nabavi, A. Chakraborty & Y. Xin, “Admm optimization strategies for wide-area oscillation monitoring in power systems under asynchronous communication delays”, *IEEE Transactions on Smart Grid*, **7**, no. 4, pp. 2123–2133, 2016.
- [65] J. Zhang, C. Chung, C. Lu, K. Men & L. Tu, “A novel adaptive wide area pss based on output-only modal analysis”, *IEEE Transactions on Power Systems*, **30**, no. 5, pp. 2633–2642, 2014.
- [66] J. Zhang, C. Chung, S. Zhang & Y. Han, “Practical wide area damping controller design based on ambient signal analysis”, *IEEE Transactions on Power Systems*, **28**, no. 2, pp. 1687–1696, 2012.
- [67] P. Zhang, D. Yang, K. Chan & G. Cai, “Adaptive wide-area damping control scheme with stochastic subspace identification and signal time delay compensation”, *IET generation, transmission & distribution*, **6**, no. 9, pp. 844–852, 2012.
- [68] S. Zhang & V. Vittal, “Design of wide-area power system damping controllers resilient to communication failures”, *IEEE J P WRS*, **28**, no. 4, pp. 4292–4300, 2013.
- [69] Y. Zhang & A. Bose, “Design of wide-area damping controllers for interarea oscillations”, *IEEE Transactions on Power Systems*, **23**, no. 3, pp. 1136–1143, 2008.
- [70] M. Zima, M. Larsson, P. Korba, C. Rehtanz & G. Andersson, “Design aspects for wide-area monitoring and control systems”, *Proceedings of the IEEE*, **93**, no. 5, pp. 980–996, 2005.